

Android 程序设计实用教程

左　军　编著

清华大学出版社

Android 程序设计实用教程

左　军　编著

清华大学出版社
北　京

内 容 简 介

本书循序渐进地对 Android 进行介绍，内容详细、充实，实例丰富、典型。本书对每个知识点都进行了详尽的描述并为每个知识点给出对应的实例说明，让读者更容易上手，对 Android 的学习会更快捷。

本书共 9 章，主要内容包括 Android 基础知识、Android 界面设计、Android 控件设计、Android 对话框与菜单、Android 视图、Android 动画、Android 绘图、Android 数据存储与共享、Android 经典应用。通过本书的学习，读者能够在较短的时间内熟悉 Android，并掌握 Android。

本书适合没有 Android 知识基础的读者，初、中级程序员以及 Android 爱好者阅读，也可供从事 Android 开发的研究人员、工作人员、高等院校相关专业的学生使用。

图书在版编目(CIP)数据

Android 程序设计实用教程/左军编著. --北京：清华大学出版社，2015
ISBN 978-7-302-39363-4

Ⅰ. ①A…　Ⅱ. ①左…　Ⅲ. ①移动终端－应用程序－程序设计－高等学校－教材　Ⅳ. ①TN929.53

中国版本图书馆 CIP 数据核字(2015)第 031614 号

责任编辑：魏江江　王冰飞
封面设计：杨　兮
责任校对：时翠兰
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　张**：28.5　　**字　　数**：694 千字
版　　次：2015 年 5 月第 1 版　　**印　　次**：2015 年 5 月第1 次印刷
印　　数：1～　000
定　　价：　.00 元

产品编号：062780-01



前　　言

Android是Google公司推出的专为移动设备开发的平台。从2007年11月5日推出以来，在短短的几年时间里就超越了称霸10年的诺基亚Symbian系统和前几年崛起的苹果iOS系统，成为全球最受欢迎的智能手机平台。应用Android不仅可以开发在手机或平板电脑等移动设备上运行的工具软件，而且可以开发2D甚至3D游戏。

从技术角度而言，Android与iPhone相似，采用WebKit浏览器引擎，具备触摸屏、高级图形显示和上网功能，用户可以在手机上查收电子邮件、搜索网址和观看视频节目等。Android手机比iPhone等其他手机更强调搜索功能，界面更丰富，可以说是一种融入了全部Web应用的平台。Android的版本包括Android 1.1、Android 1.5、Android 1.6、Android 2.0……当前的最新版本是4.4.x。随着版本的更新，从最初的触屏到现在的多点触摸，从普通的联系人到现在的数据同步，从简单的GoogleMap到现在的导航系统，从基本的网页浏览到现在的HTML5，都说明Android已经逐渐稳定，而且功能越来越大。此外，Google平台不仅支持Java、C、C++等主流编程语言，还支持Ruby、Python等脚本语言，甚至Google专为Android的应用开发推出了Simple语言，这使得Android有着非常广泛的开发群体。

虽然Android是优秀的移动操作系统，但是其程序开发的学习之旅却很艰难，最大的困难就是相关资料的缺乏。Android是完全开源的，但不是每个程序设计人员都有时间和精力去研究它的源代码。Google提供的主要学习资料就是Android SDK文档。SDK文档对于开发人员了解Android程序设计有很大的帮助，但并没有系统地讲解Android程序设计的相关技术。针对所存在的问题，本书就此诞生。

本书通过对Android程序设计基础知识和基本技能进行全面系统的讲解，使读者能够轻松地掌握Android程序设计的基本知识和技能，尽量减少在Android程序设计入门阶段的摸索和徘徊，为学习Android程序高级技术打下基础。本书具有以下特色：

1. 结构合理

从用户的实际出发，科学安排知识内容，内容由浅入深，叙述清晰，并附加相应的实例进行操作，具有很强的知识性和实用性，反映了当前Android网络开发技术的发展和应用水平。

2. 通俗易懂

内容条理清晰、语言简练，可以帮助读者快速掌握每个知识点；每个部分既相互连贯又自成一体，使读者既可以按照本书编排的章节进行学习，也可以根据自己的需求对某一章节进行针对性学习。

3. 实用性强

本书彻底摒弃枯燥的理论和简单的操作，注重实用性和可操作性，将Android网络开发技术的理论融合到实际的操作环境中，使用户在掌握相关操作技能的同时，还能够学习到相

应的开发知识。

4. 实例丰富

书中的实例应用全面，涵盖了 Android 所能触及的领域。实例代码翔实、规范工整，且代码注释得当。

5. 图文并茂

针对没有接触过 Android 的读者，本书对相关概念一般会插入对应的图片做说明，同时几乎对每一个知识点实例都给出相应的运行效果图，这样对读者掌握这一知识点起到了很大的帮助作用。

本书共分为 9 章，其主要内容为：

第 1 章进入 Android，主要包括 Android 基本知识、Android 开发环境搭建、Android 应用组成以及 Android 模拟器操作等内容。

第 2 章 Android 界面设计，主要包括 Android 的 UI 界面、Android 布局管理器以及自定义 View 等内容。

第 3 章 Android 控件设计，主要包括 Android 文本类控件、按钮控件、编辑框控件以及条类控件等内容。

第 4 章 Android 对话框与菜单，主要包括 Android 对话框类型、Android 各类菜单等内容。

第 5 章 Android 视图，主要包括 Android 图像视图、网格视图、画廊视图、多页视图等内容。

第 6 章 Android 动画，主要包括 Android 帧动画、补间动画、动画渲染器以及动画组件等内容。

第 7 章 Android 绘图，主要包括 Android 2D 绘图及 3D 绘图等内容。

第 8 章 Android 数据存储与共享，主要包括 SharedPreferences 存储数据、File 存储数据、SQLite 存储数据以及 ContentProvider 数据共享等内容。

第 9 章 Android 经典应用，主要包括 Android 多媒体技术、无线网络、通信、定位等内容。

本书适合对象主要有以下几类：

- Android 入门级开发人员；
- 初、中级程序员；
- 培训班学生；
- Android 爱好者；
- 从事 Android 开发的研究人员和工作人员；
- 高等院校相关专业的学生。

本书主要由左军编写，此外参加编写的还有刘超、邓俊辉、梁朗星、李旭波、张棣华、刘泳、邓耀隆、梁志成和周品。

由于作者的水平有限，加之时间紧凑，书中难免存在不足之处，敬请广大读者批评指正。

编　者

2015 年 1 月

目　　录

第 1 章　进入 Android

在快速发展的移动开发领域中，以 Android 的发展最为迅猛。仅仅短短的几年，就撼动了诺基亚的霸主地位。通过在线市场，Android 的程序员不仅能向全世界贡献自己的程序，而且也能通过销售获得不菲的收入。

1.1　揭开 Android 的面纱

Android 是一种基于 Linux 的自由及开放源代码的操作系统，主要使用于移动设备，例如智能手机和平板电脑，由 Google 公司和开放手机联盟领导及开发。第一部 Android 智能手机发布于 2008 年 10 月。Android 逐渐扩展到平板电脑及其他领域上，例如电视、数码相机、游戏机等。2011 年的第一季度，Android 在全球的市场份额首次超过塞班系统，跃居全球第一。2013 年的第四季度，Android 平台手机的全球市场份额已经达到 78.1%。2014 年 9 月 24 日，谷歌开发的操作系统 Android 在迎来了 6 岁生日时，全世界采用这款系统的设备数量已经达到 15 亿台。

2014 年的第一季度 Android 平台已占所有移动广告流量来源的 42.8%，首度超越 iOS。但运营收入还不及 iOS。

1.1.1　Android 体系结构

Android 系统是以 Linux 系统为基础的，Google 按照功能特性将其划分为 4 层，自下而上分别是 Linux 内核、中间件、应用程序框架和应用程序，如图 1-1 所示。

1. 应用程序

Android 系统内置了一些常用的应用程序，包括 Home 视图、联系人、电话、浏览器等。这些应用程序和用户自己编写的应用程序是完全并列的，同样都是采用 Java 语言编写的。而且，用户可以根据需要增加自己的应用程序，或者替换系统自带的应用程序。

2. 应用程序框架

应用程序框架提供了程序开发人员的接口，这是与 Android 程序员直接相关的部分，开发者可以用它开发应用程序。

- 丰富而又可扩展的视图（Views）：可以用来构建应用程序，它包括列表（lists）、网格（grids）、文本框（text boxes）、按钮（buttons），甚至可嵌入的 Web 浏览器。
- 内容提供器（Content Providers）：使得应用程序可以访问另一个应用程序的数据（如联系人数据库），或者共享它们自己的数据。
- 资源管理器（Resource Manager）：提供非代码资源的访问，如本地字符串、图形、布

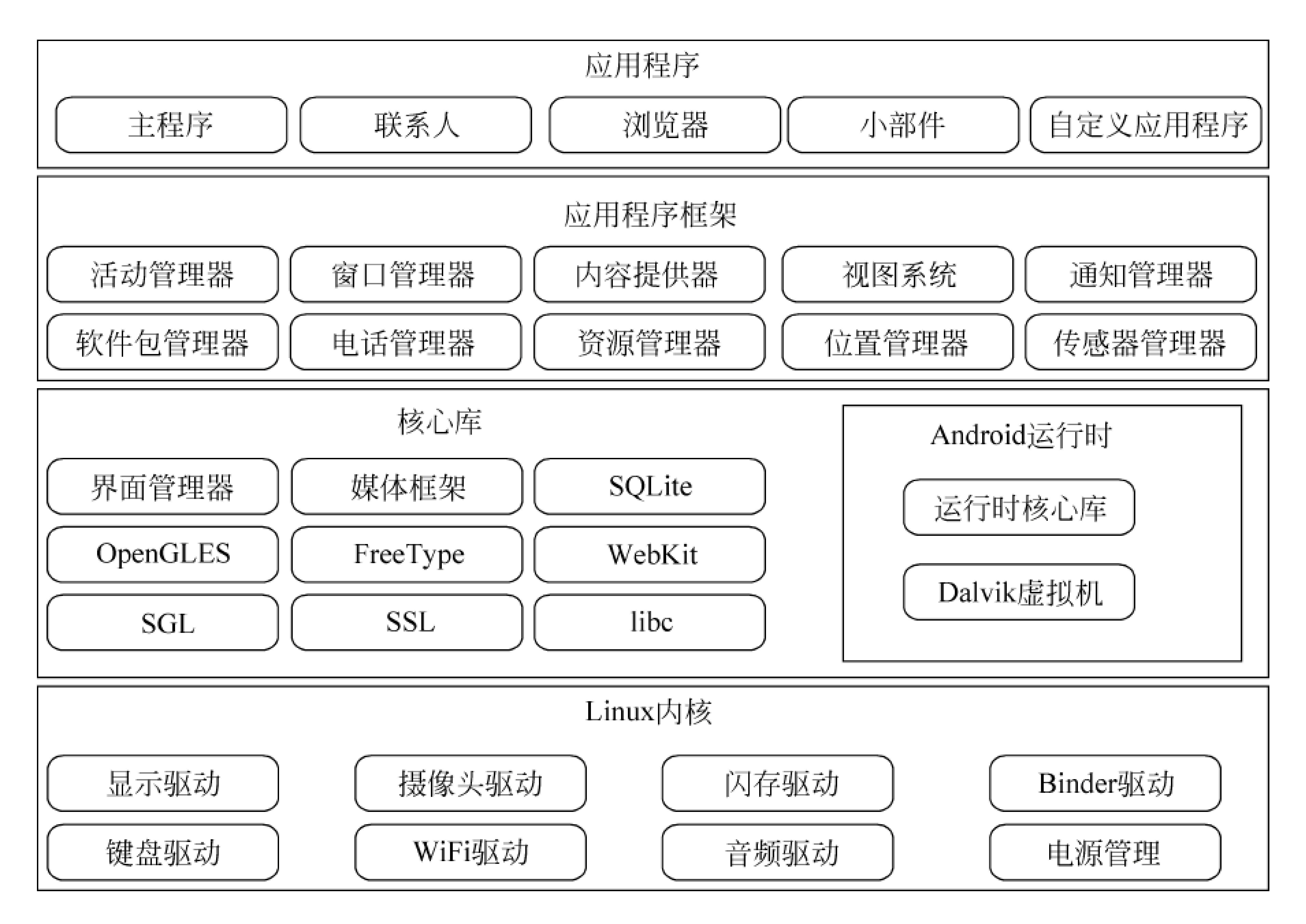


图 1-1　Android 系统框架图

局文件(layoutfiles)。

- 通知管理器(Notification Manager):使得应用程序可以在状态栏中显示自定义的提示信息。
- 活动管理器(Activity Manager):用来管理应用程序生命周期并提供常用的导航回退功能。

3. 中间件

中间件包括核心库(libraries)和 Android 运行时环境(Android runtime)两部分。

1) 核心库

核心库中主要包括一些 C/C++核心库,方便开发者进行应用的开发。

- 系统 C 库(libc):专门为基于 embedded linux 的设备定制的。
- 媒体库:支持多种常用的音频、视频格式回放和录制,同时支持静态图像文件。编码格式包括 MPEG4、H.264、MP3、AAC、AMR、JPG、PNG。
- SurfaceManager:对显示子系统的管理,并且为多个应用程序提供了 2D 和 3D 图层的无缝融合。
- Webkit/LibWebCore:Web 浏览引擎,支持 Android 浏览器和一个可嵌入的 Web 视图。
- SGL:底层的 2D 图形引擎。
- 3D libraries:基于 OpenGL ES 1.0 APIs 实现的 3D 引擎。
- FreeType:位图(bitmap)和矢量(vector)字体显示。
- SQLite:轻型关系型数据库引擎。

2) Android 运行时环境

Android 运行时环境主要包括以下两部分。

(1) Android 核心库:提供了 Java 库的大多数功能。

(2) Dalvik 虚拟机：依赖于 Linux 内核的一些功能，例如线程机制和底层内存管理机制。同时虚拟机是基于寄存器的，Dalvik 采用简练、高效的 byte code 格式运行，它能够在低资耗和没有应用相互干扰的情况下并行执行多个应用，每一个 Android 应用程序都在它自己的进程中运行，都拥有一个独立的 Dalvik 虚拟机实例。Dalvik 虚拟机中可执行文件为 .dex 文件，该格式文件针对小内存使用做了优化。所有的类都经由 Java 编译器编译，然后通过 SDK 中的 dx 工具转化成.dex 格式由虚拟机执行。

4. Linux 内核

Android 平台运行在 Linux 2.6 之上，其 Linux 内核部分相当于手机硬件层和软件层之间的一个抽象层。Android 的内核提供了显示驱动、摄像头驱动、闪存驱动、键盘驱动、WiFi 驱动、音频驱动和电源管理等多项功能。此外，Android 为了让 Android 程序可以用于商业目的，将 Linux 系统中受 GNU 协议约束的部分进行了取代。

1.1.2 Android 自身特性

Android 是一种开源操作系统，其在手机操作系统领域的市场占有率已经超过了 50%，是什么原因让 Android 操作系统如此受欢迎呢？因为有其自身的几大优势。

1. 开放性

在优势方面，Android 平台首先就是开放性，开放的平台允许任何移动终端厂商加入到 Android 联盟中来。显著的开放性可以使其拥有更多的开发者，随着用户和应用的日益丰富，一个崭新的平台也将很快走向成熟。

开放性对于 Android 的发展而言，有利于积累人气，这里的人气包括消费者和厂商，而对于消费者来讲，最大的受益正是丰富的软件资源。开放的平台也会带来更大竞争，如此一来，消费者将可以用更低的价位购得心仪的手机。

2. 不受束缚

在过去很长的一段时间，特别是在欧美地区，手机应用往往受到运营商的制约，使用什么功能接入什么网络，几乎都受到运营商的控制。自从 2007 年 iPhone 上市后，用户可以更加方便地连接网络，运营商的制约减少。随着 EDGE、HSDPA 这些 2G 至 3G 移动网络的逐步过渡和提升，手机随意接入网络已不是运营商口中的笑谈。

3. 丰富的硬件

这一点还是与 Android 平台的开放性相关，由于 Android 的开放性，众多的厂商会推出千奇百怪、功能特色各具的多种产品。功能上的差异和特色，却不会影响到数据同步，甚至软件的兼容，如同从诺基亚 Symbian 风格手机一下改用苹果 iPhone，同时还可将 Symbian 中优秀的软件带到 iPhone 上使用，联系人等资料更是可以方便地转移。

4. 方便开发

Android 平台提供给第三方开发商一个十分宽泛、自由的环境，不会受到各种条条框框的阻碍，可想而知，会有多少新颖别致的软件诞生。但也有其两面性，血腥、暴力、情色方面的程序和游戏如何控制正是留给 Android 的难题之一。

5. Google 应用

在互联网的 Google 已经走过 10 年的历史，从搜索巨人到全面的互联网渗透，Google 服务如地图、邮件、搜索等已经成为连接用户和互联网的重要纽带，而 Android 平台手机将

无缝结合这些优秀的 Google 服务。

1.1.3 Android 应用组件

Android 开发四大组件分别是活动(Activity):用于表现功能。服务(Service):后台运行服务,不提供界面呈现。广播接收器(BroadcastReceiver):用于接收广播。内容提供商(Content Provider):支持在多个应用中存储和读取数据,相当于数据库。

1. 活动

在 Android 中,Activity 是所有程序的根本,所有程序的流程都运行在 Activity 之中,Activity 可以算是开发者遇到的最频繁,也是 Android 当中最基本的模块之一。在 Android 的程序当中,Activity 一般代表手机屏幕的一屏。如果把手机比作一个浏览器,那么 Activity 就相当于一个网页。在 Activity 中可以添加一些 Button、Checkbox 等控件,可以看到 Activity 概念和网页的概念类似。

一般一个 Android 应用是由多个 Activity 组成的。这多个 Activity 之间可以进行相互跳转,例如,单击一个 Button 按钮后,可能会跳转到其他的 Activity。和网页跳转稍微有些不一样的是,Activity 之间的跳转有可能返回值,例如,从 Activity A 跳转到 Activity B,那么当 Activity B 运行结束的时候,有可能会给 Activity A 一个返回值。这样做在很多时候是相当方便的。

Android 4 种的 Activity 加载模如图 1-2 所示。

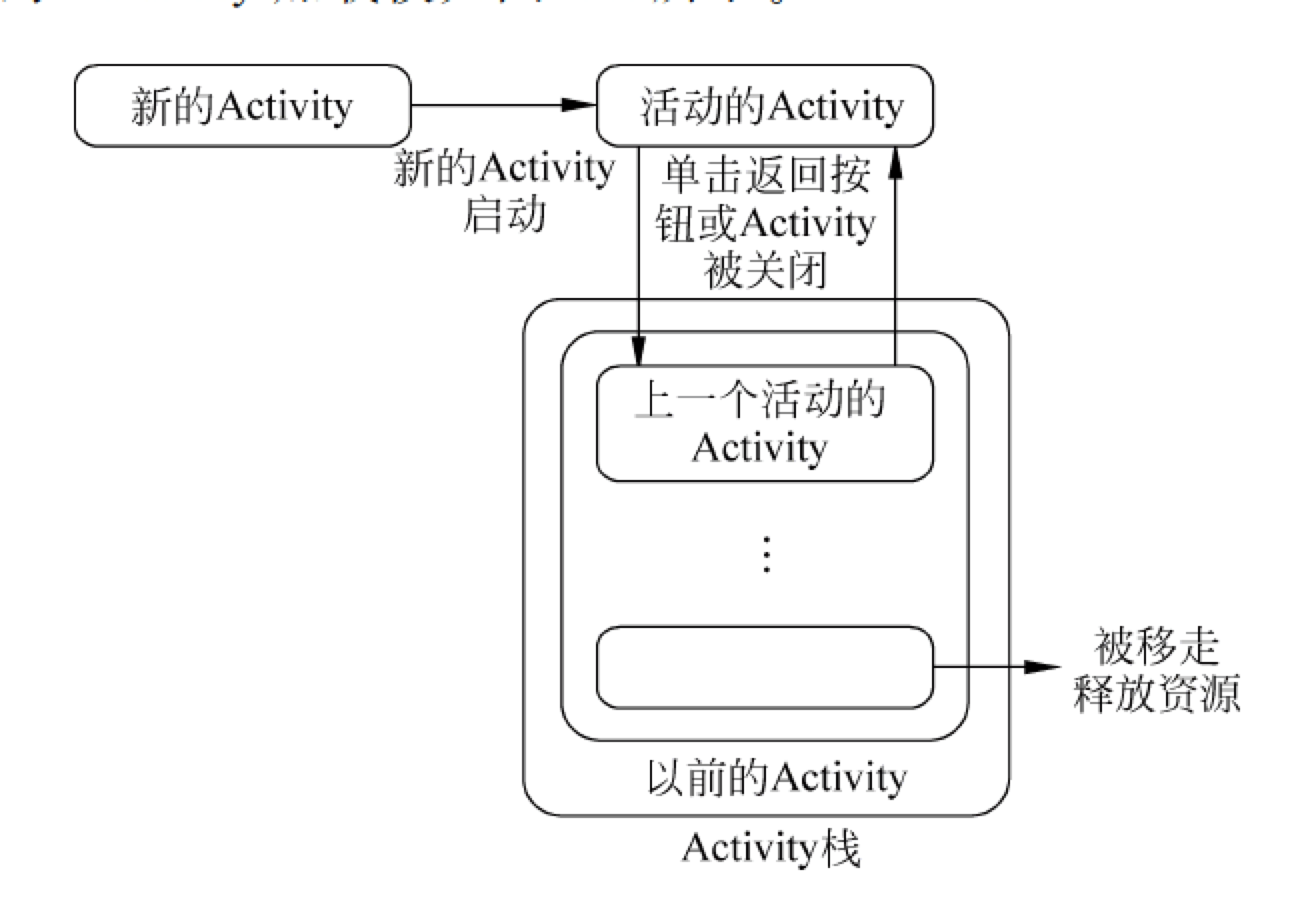


图 1-2 Android 4 种 Activity 加载模流程图

当打开一个新的屏幕时,之前的一个屏幕会被置为暂停状态,并且压入历史堆栈中。用户可以通过回退操作返回到以前打开过的屏幕。可以选择性地移除一些没有必要保留的屏幕,因为 Android 会把每个应用的开始到当前的每个屏幕保存在堆栈中。

2. 服务

Service 是 Android 系统中的一种组件,它跟 Activity 的级别差不多,但是它不能自己运行,只能在后台运行,并且可以和其他组件进行交互。Service 是没有界面的长生命周期的代码。Service 是一种程序,它可以运行很长时间,但是它却没有用户界面。这么说有点枯燥,来看一个例子。打开一个音乐播放器的程序,这个时候如果想上网了,那么,打开 Android 浏览器,这个时候虽然已经进入了浏览器这个程序,但是,歌曲播放并没有停止,而

是在后台继续一首接着一首的播放。其实这个播放就是由播放音乐的 Service 进行控制。当然这个播放音乐的 Service 也可以停止，例如，当播放列表里边的歌曲都结束了，或者用户单击了停止音乐播放的快捷键等。Service 可以在和多场合的应用中使用，例如播放多媒体的时候用户启动了其他 Activity，这个时候程序要在后台继续播放，例如检测 SD 卡上文件的变化，再或者在后台记录地理信息位置的改变，等等。

开启 Service 有以下两种方式。

（1）Context. startService()：Service 会经历 onCreate→onStart（如果 Service 还没有运行，则 Android 先调用 onCreate()，然后调用 onStart()；如果 Service 已经运行，则只调用 onStart()，所以一个 Service 的 onStart 方法可能会重复调用多次）；StopService 的时候直接 onDestroy，如果是调用者自己直接退出而没有调用 StopService 的话，Service 会一直在后台运行。该 Service 的调用者再启动后可以通过 StopService 关闭 Service。注意，多次调用 Context. startservice()不会嵌套（即使会有相应的 onStart()方法被调用），所以无论同一个服务被启动了多少次，一旦调用 Context. stopService()或者 StopSelf()，它都会被停止。补充说明：传递给 StartService()的 Intent 对象会传递给 onStart()方法。调用顺序为：onCreate →onStart（可多次调用）→onDestroy。

（2）Context. bindService()：Service 会经历 onCreate()→onBind()，onBind 将返回给客户端一个 IBind 接口实例，IBind 允许客户端回调服务的方法，例如得到 Service 运行的状态或其他操作。这个时候调用者（Context，例如 Activity）会和 Service 绑定在一起，而 Context 退出，接着 Service 就会调用 onUnbind→onDestroyed 退出，所谓绑定在一起即为"共存亡"。

3. 广播接收器

在 Android 中，Broadcast 是一种广泛运用的在应用程序之间传输信息的机制。而 BroadcastReceiver 是对发送出来的 Broadcast 进行过滤接收并响应的一类组件。可以使用 BroadcastReceiver 来让应用对一个外部的事件做出响应。这是非常有意思的，例如，当电话呼入这个外部事件到来的时候，可以利用 BroadcastReceiver 进行处理。例如，当下载一个程序成功完成的时候，仍然可以利用 BroadcastReceiver 进行处理。BroadcastReceiver 不能生成 UI，也就是说对于用户来说不是透明的，用户是看不到的。BroadcastReceiver 通过 NotificationManager 来通知用户这些事情发生了。BroadcastReceiver 既可以在 AndroidManifest. xml 中注册，也可以在运行时的代码中使用 Context. registerReceiver()进行注册。只要是注册了，当事件来临的时候，即使程序没有启动，系统也会在需要的时候启动程序。各种应用还可以通过使用 Context. sendBroadcast()将它们自己的 Intent Broadcasts 广播给其他应用程序。

4. 内容提供商

Content Provider 是 Android 提供的第三方应用数据的访问方案。

在 Android 中，对数据的保护是很严密的，除了放在 SD 卡中的数据，一个应用所持有的数据库、文件等内容，都是不允许其他应用直接访问的。Android 当然不会真的把每个应用都做成一座"孤岛"，它为所有应用都准备了一扇窗，这就是 Content Provider。如果应用想对外提供的数据，可以通过派生 Content Provider 类，封装成一枚 Content Provider，每个 Content Provider 都用一个 uri 作为独立的标识，形如 content://com. xxxxx。所有应用看

着像 REST 的样子，但实际上，它比 REST 更为灵活。和 REST 类似，uri 也可以有两种类型，一种是带 id 的，另一种是列表的，但实现者不需要按照这个模式来做，给 id 的 uri 也可以返回列表类型的数据。

1.2 Android 开发环境

“工欲善其事，必先利其器”，在学习 Android 开发前，必须先熟悉并搭建它所需要的开发环境。

1.2.1 Android 系统需求

使用 Android SDK 进行开发时有其所必需的硬件和软件需求。对于硬件方面，要求 CPU 和内存尽量大。Android SDK 全部下载大概需要占用 4.5 GB 硬盘空间。由于开发过程中需要反复重启模拟器，而每次重启都会消耗几分钟的时间(视机器配置而定)，因此使用高配置的机器能节约不少时间。

支持 Android SDK 的操作系统及其要求如表 1-1 所示。

表 1-1 Android SDK 对操作系统的要求

操作系统	要求
Windows	Windows XP(32 位)
	Vista(32 或 64 位)
	Windows 7(32 位或 64 位)
Mac OS	10.5.8 或更新(仅支持 x86)
Linux(在 Ubuntu 的 10.04 版测试)	需要 GNU C Library(glibc)2.7 或更新 在 Ubuntu 系统上，需要 8.04 版或更新 64 位版本必须支持 32 位应用程序

对于开发环境，除了常用的 Eclipse IDE，还可以使用 Intelli J IDEA 进行开发。对于 Eclipse 在下载 Android SDK 时就自带相兼容的版本。

1.2.2 Android 环境搭建

在 Windows 平台上，搭建 Android 开发环境，首先下载并安装与开发环境相关的软件资源，这些资源主要包括 JDK、Eclipse、Android SDK 和 Development Tools 插件(ADT 插件)。

在 Android 平台上，所有应用程序都是使用 Java 语言来编写的，所以要安装 Java 开发包 JDK(Java SE Development Kit)，JDK 是 Java 开发时所必需的软件开发包。

安装 JDK 的过程比较简单，运行该程序后，根据安装提示选择安装路径，将 JDK 安装到指定的文件夹即可，默认安装目标为 C:\Program Files\Java\jdk1.6.0_10(jdk-6u10-rc2-bin-b32-windows-i586-p-12_sep_2008)。

JDK 安装完毕后，进一步要设置 Java 的环境变量，即设置 bin 和 lib 文件夹的路径。其操作步骤如下(在计算机操作系统为 Windows 7 的环境下)：

(1) 右击“计算机”,在弹出的快捷菜单中选择“属性”选项,在弹出的“系统”对话框中,单击“高级系统设置”按钮,弹出“系统属性”对话框,如图 1-3 所示。

(2) 在“系统属性”对话框的“高级”选项卡中,单击“环境变量”按钮,弹出“环境变量”对话框,如图 1-4 所示。



图 1-3 “系统属性”对话框

图 1-4 “环境变量”对话框

(3) 选中“系统变量”区域的 PATH 变量,单击“编辑”按钮,弹出“编辑系统变量”对话框,如图 1-5 所示。

(4) 在该对话框的“变量值”文本框中添加 C:\Program Files\Java\jdk1.6.0_10\bin,然后单击“确定”按钮即可完成设置。这样即设置了 bin 文件夹的路径。

(5) 在“环境变量”对话框的“系统变量”区域中,单击“新建”按钮,弹出“新建系统变量”对话框,如图 1-6 所示。

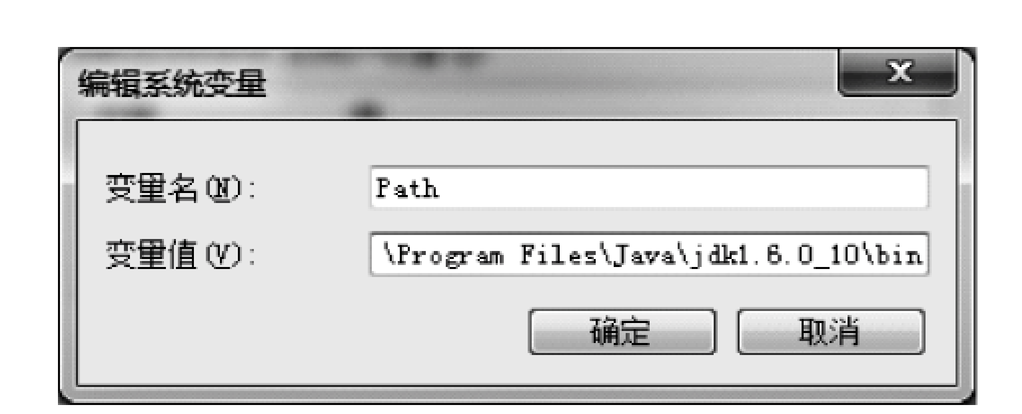


图 1-5 环境变量 Path 设置



图 1-6 新建环境变量 classpath

(6) 在图 1-6 中的“变量名”右侧文本框中输入 classpath,在“变量值”右侧文本框中输入 C:\Program Files\Java\jdk1.6.0_10\lib,即可设置 lib 文件夹的路径。

完成以上操作后,一个典型的 Java 开发环境便设置好了。在正式开始下一步前先验证 Java 开发环境的设置是否成功。

在 Windows 7 系统中单击“开始”按钮,在弹出的窗口中选择“运行”,在运行框中输入 cmd 并确定,即可打开 CMD 窗口,在窗口中输入 java-version,则可显示所安装的 Java 版本信息,如图 1-7 所示。



图 1-7　JDK 安装成功页面

1.2.3　Eclipse 环境

从 Android 4.4 版本开始，下载的 Android 软件包中包括 Eclipse 软件，在官方网站下载相应 Android 软件包并解压后即可看到 Eclipse 软件启动器，双击该软件，打开效果如图 1-8 所示。



图 1-8　Eclipse 启动界面

启动 Eclipse 开发环境桌面，将会看到选择工作空间的提示，如图 1-9 所示。

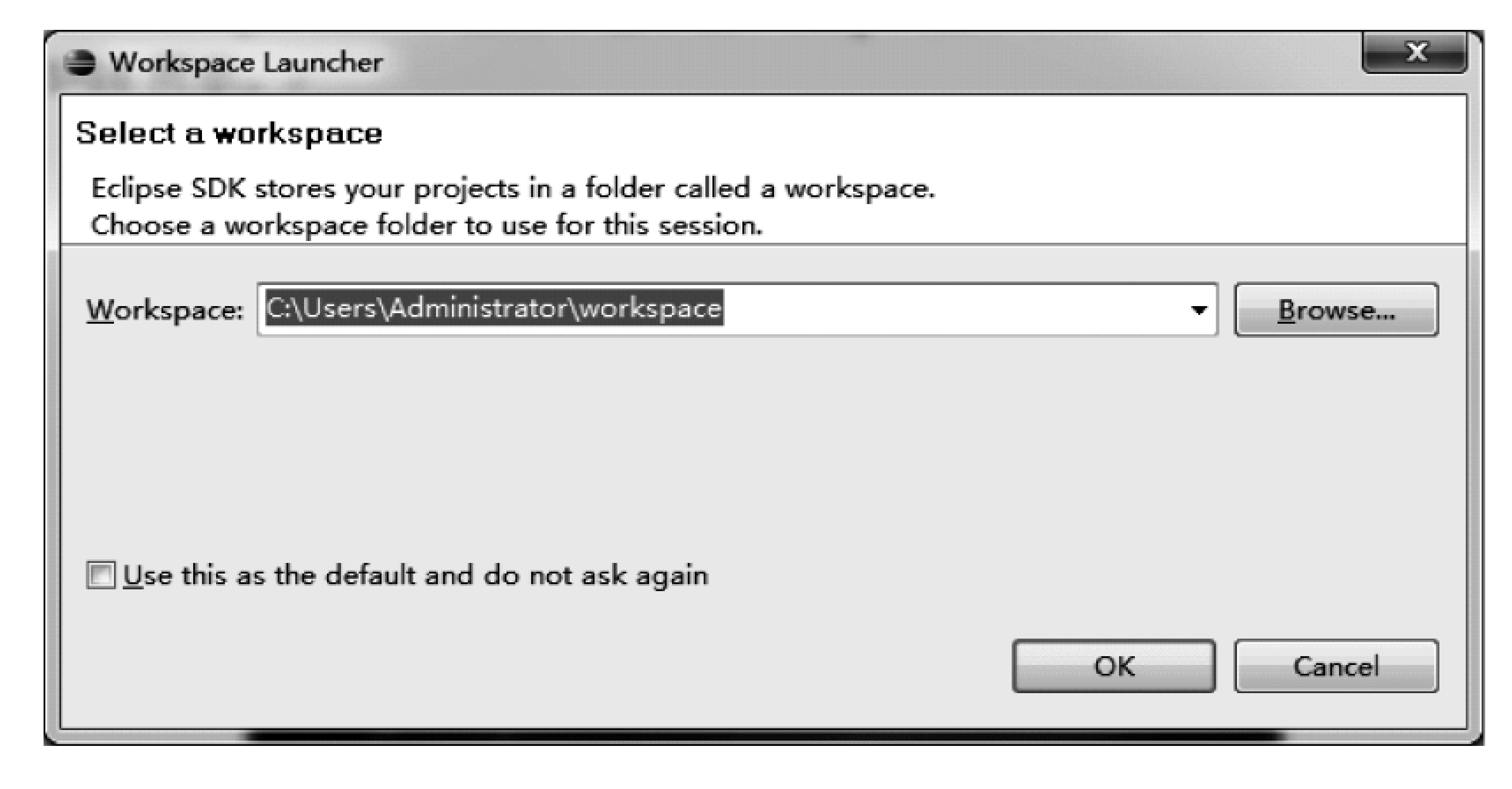


图 1-9　选择工作空间

之后单击图 1-9 中的 OK 按钮，即完成 Eclipse 的安装，系统进入 Eclipse 初始欢迎界面，如图 1-10 所示。之后单击图 1-10 左上角的“欢迎”按钮，即可进入 Eclipse 的开发环境界面，如图 1-11 所示。



图 1-10　Eclipse 欢迎界面

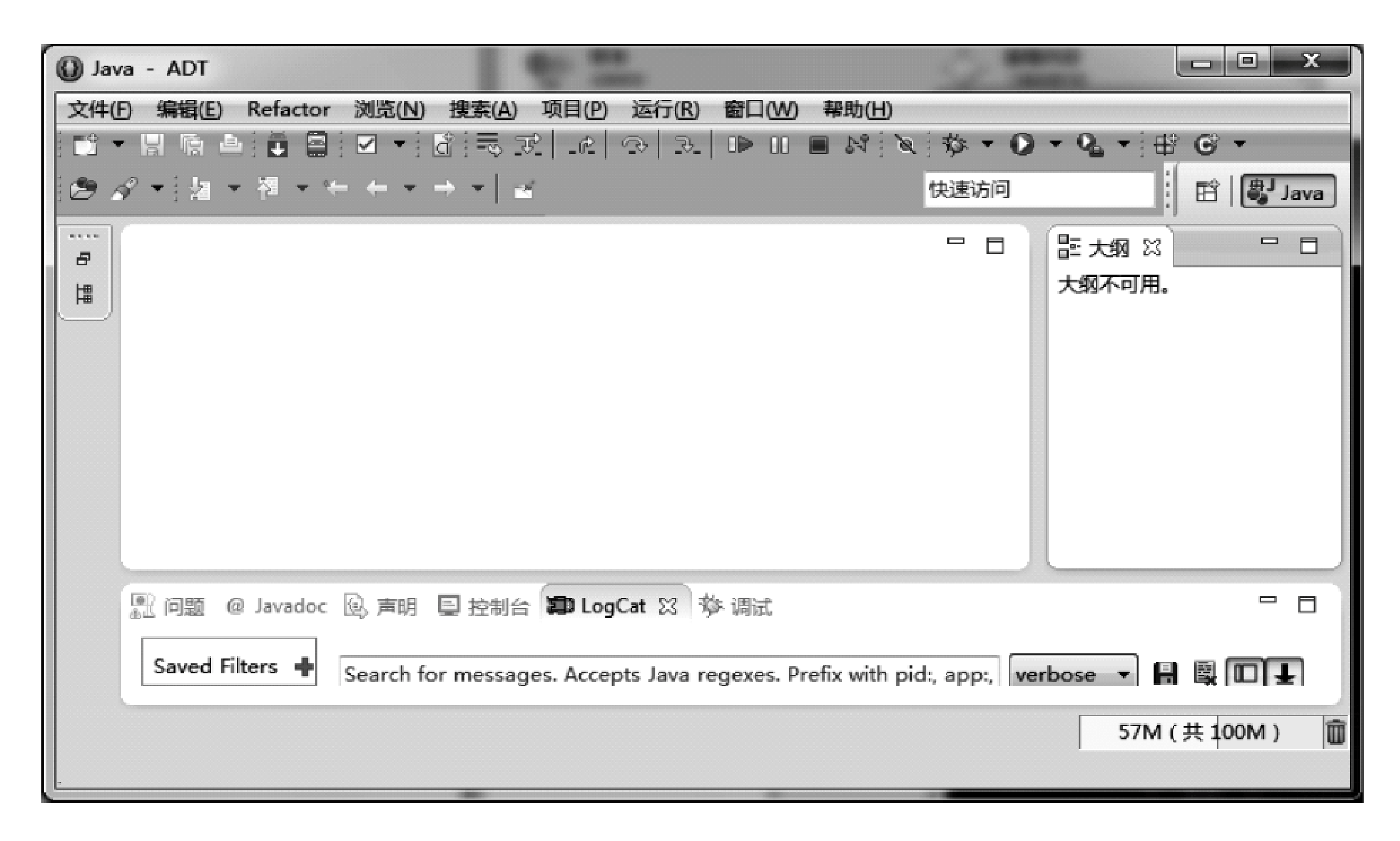


图 1-11　Eclipse 的开发界面

1.2.4　Android 的 ADK

（1）单击图 1-11 中的 [icon] 快捷按钮，程序将自动检测是否有更新的 SDK 数据包可下载，如图 1-12 所示。

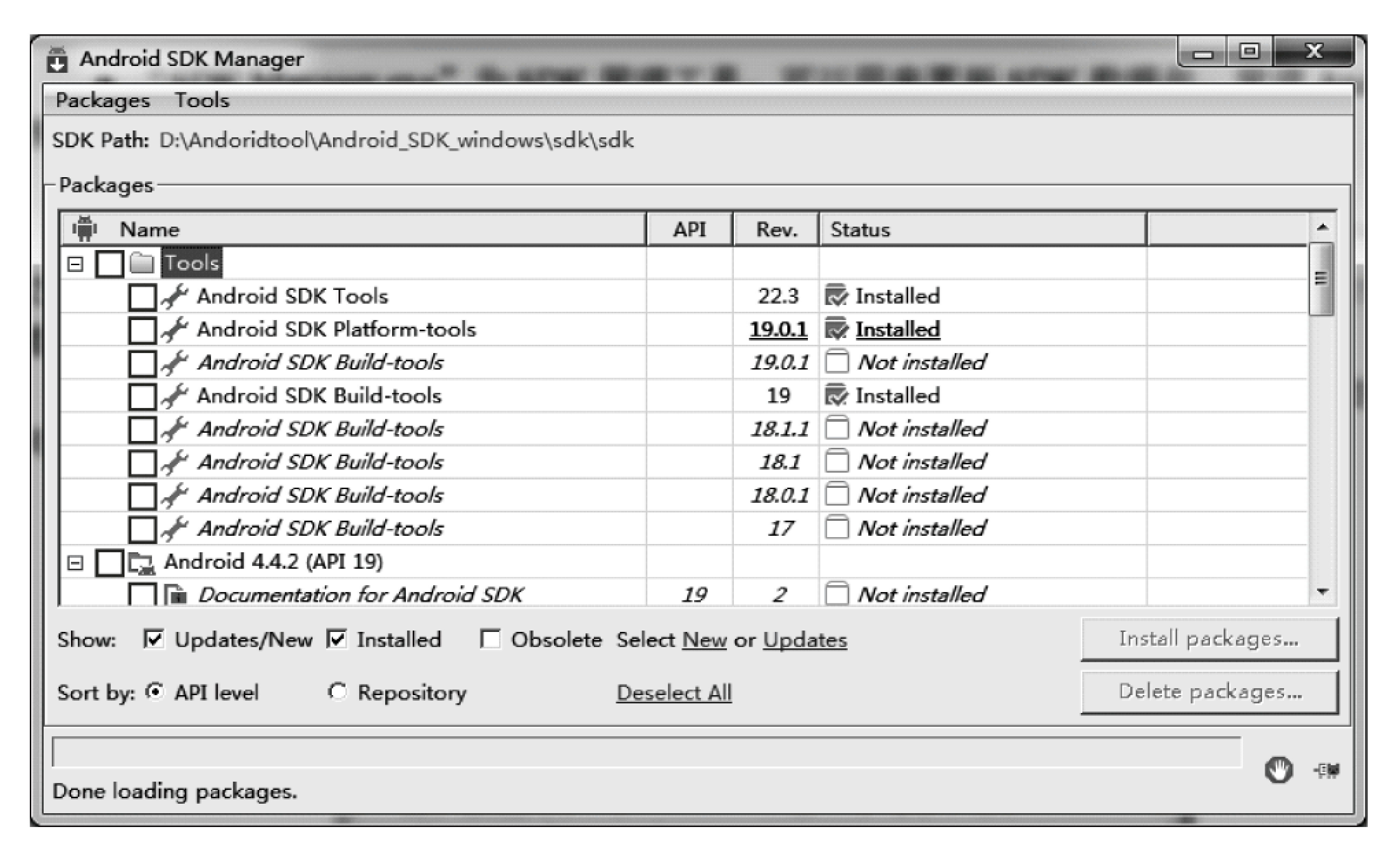


图 1-12　运行 SDK Manager. exe 执行文件

(2) 对于所要更新的内容,如果只要尝试一下 Android 4.4.2,那么只选择 Android 4.4.2 (API 19)然后单击 Install X packages 按钮来安装就可以了。如果要在此 SDK 上开发应用程序和游戏应用,那么需要接受并遵守所有许可内容(Accept All),并单击 Install 按钮。

(3) 之后将 SDK tools 目录的完整路径设置到系统变量 Path 中,这样便于在后面调用 Android 命令时,无须输入全部的绝对路径。设置系统变量 Path 的方法与 JDK 的环境变量值操作一致,在 Path 环境变量的“变量值”文本框中添加;D:\Androidtool\Android_SDK_windows\sdk\sdk\tools;即可,如图 1-13 所示。



图 1-13　设置 Android SDK 环境变量

最后检查 Android SDK 是否安装成功,是否能够正常运行。在 Windows 7 系统中单击“开始”按钮,在弹出的窗口中选择“运行”,在运行框中输入 cmd 并确定,即可打开 CMD 窗口,在窗口中输入 android-h,则可显示所安装的 Android SDK 信息,如图 1-14 所示。

1.2.5　Android 的 AVD

AVD 全称为 Android Virtual Device,即 Android 运行的虚拟设备,它是 Android 的模拟器识别。建立的 Android 要运行,必须创建 AVD,每个 AVD 上可以配置很多的运行项目。创建 AVD 时,可以配置的选项有模拟影像大小、触摸屏、轨迹球、摄像头、屏幕分辨率、键盘、GSM、GPS、Audio 录放、SD 卡支持、缓存大小等。

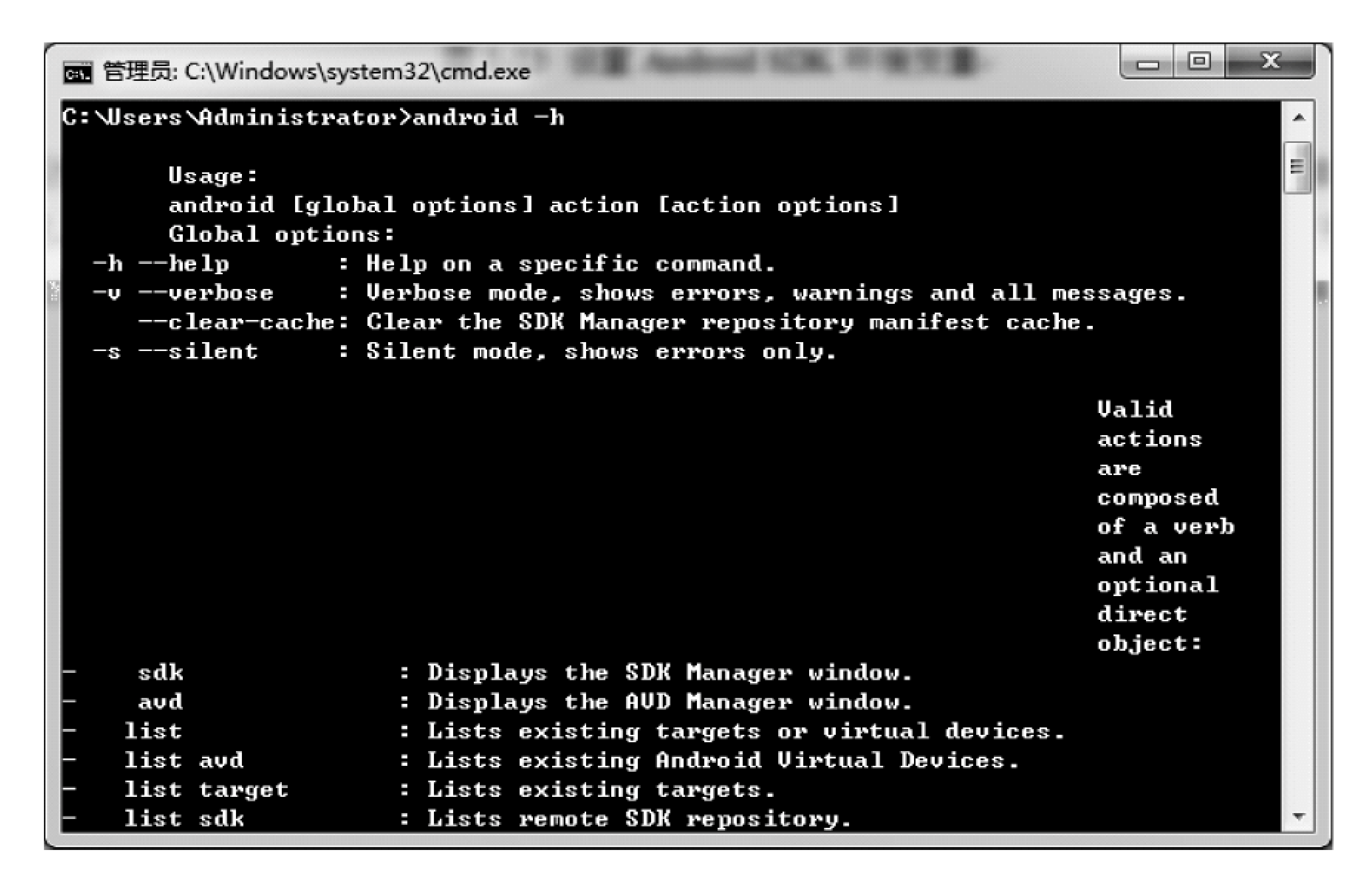


图 1-14　Android SDK 安装成功信息

(1) 单击图 1-11 中的 快捷按钮，即可启动 Android AVD，在弹出如图 1-15 所示的 Android Virtual Device Manager 窗口。

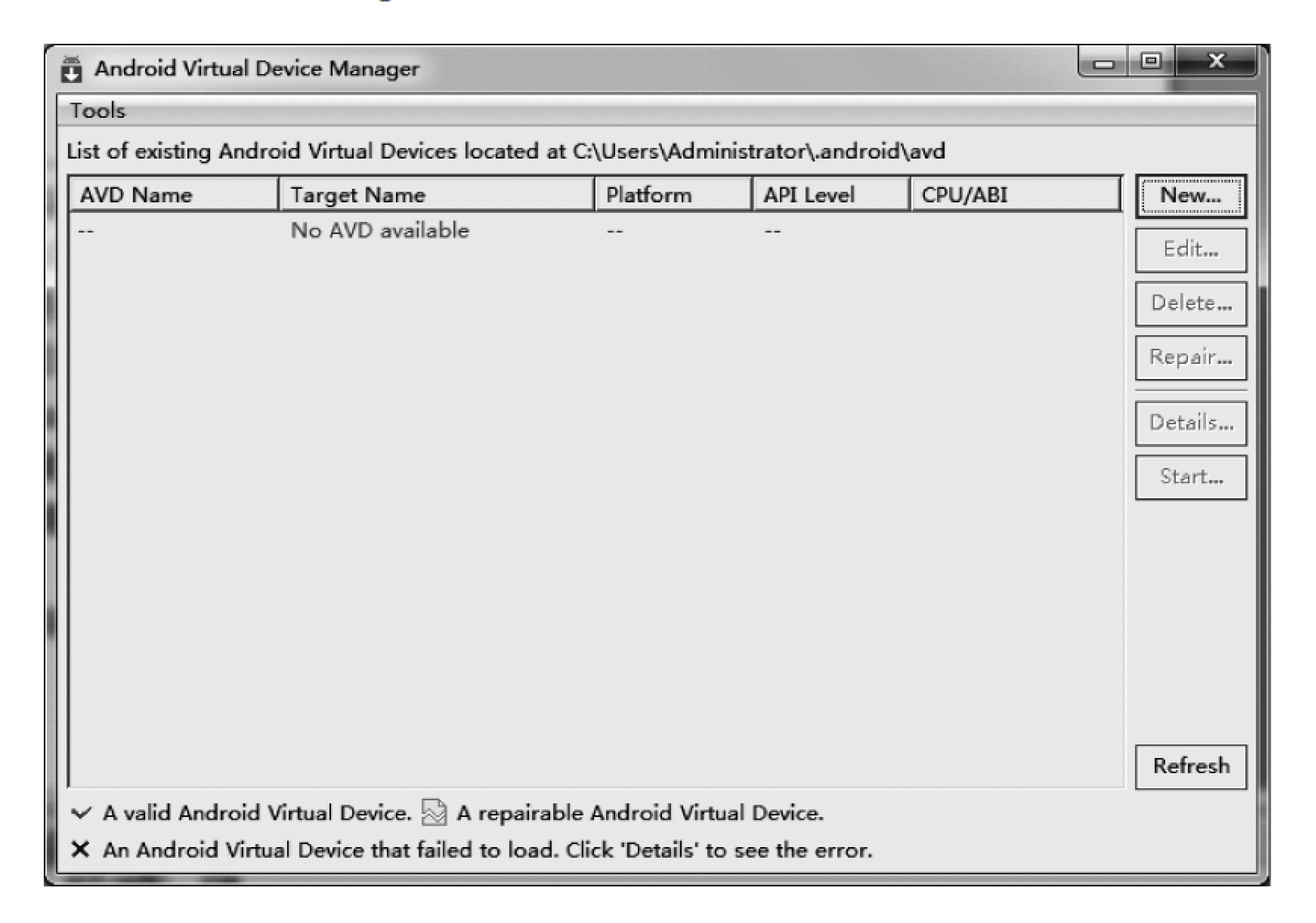


图 1-15　AVD Manager. exe 界面

(2) 单击图 1-15 右侧的 New 按钮，弹出一个新的 Create new Android Virtual Device (AVD)对话框，如图 1-16 所示。在该对话框中可以设置模拟器的配置，包括如下几项。

- AVD Name：创建 AVD 的名称。可以在文本框中输入所要创建的 AVD 的名称，注意名称中不能有空格符。
- Target：选择 Android 版本和 API 的等级。单击右边的下拉按钮，选择相应的 Android 版本和 API 的等级。

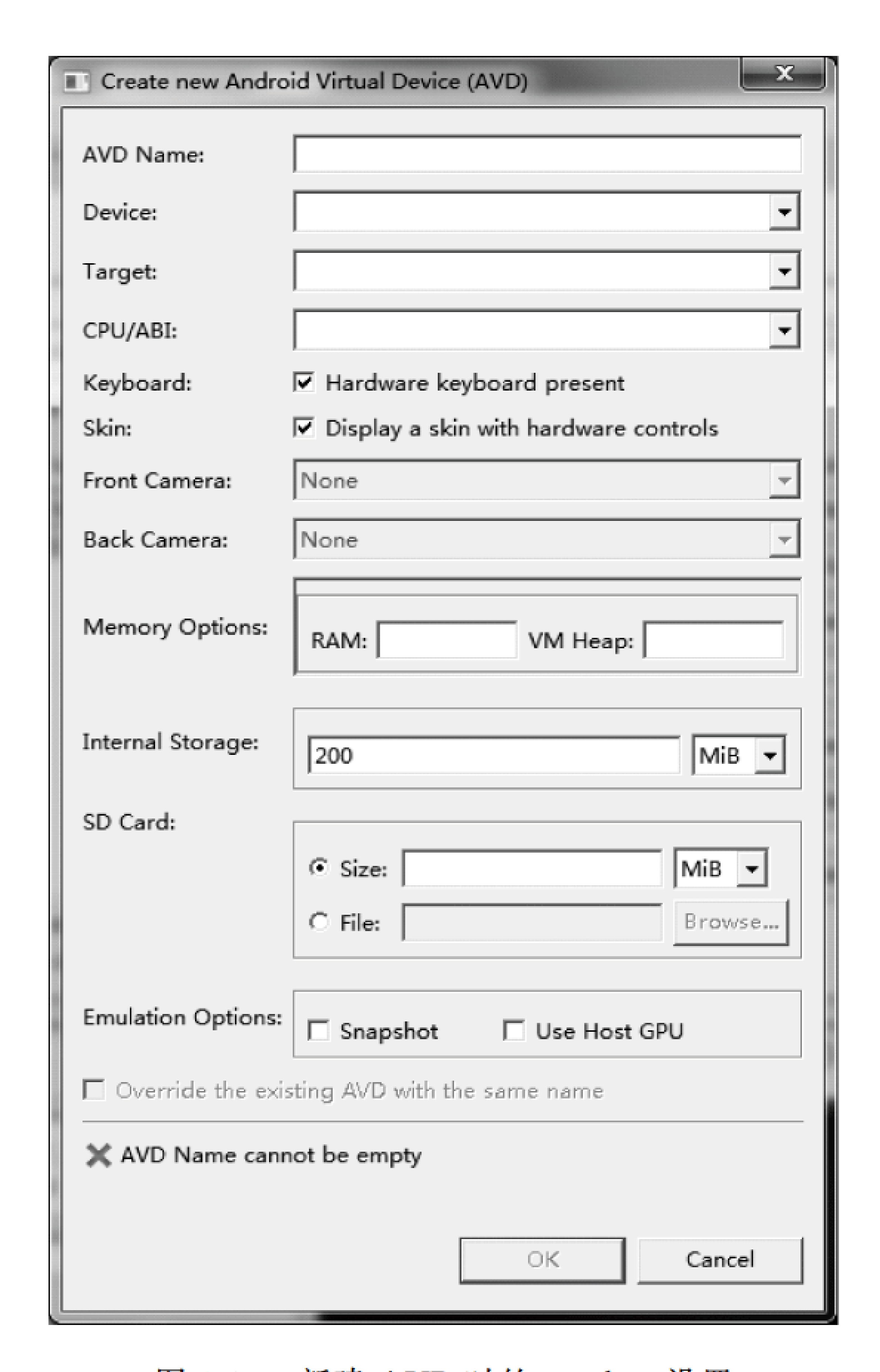


图 1-16　新建 AVD 时的 emulate 设置

- SD Card：设置 SD 卡。在 Size 文本中指定 SD 卡大小。另外，也可以在 File 文本框设置已有的 SD 卡镜像文件的路径。
- Skin：设置模拟器的外观和屏幕分辨率。单击 Built-in 右边的下拉按钮，可以选择默认的 HVGA(320×480)、QVGA(240×320)、WVGA(480×800 或 480×854)、WQVGA(240×400 或 240×320)几种，在此选择默认的 HVGA(320×480)。另外，单击 Resolution 项，还可以自定义分辨率。不同版本的 Android 所设置的 Skin 参数有所不同。
- Hardware：设置模拟器支持的硬件设备的属性，包括影像大小、触摸屏、轨迹球、摄像头、屏幕分辨率、键盘、GSM、GPS、Audio 录放、SD 卡支持、缓存区大小等。单击该区域右边的 New 按钮，在弹出的对话框中可以设置各项的属性。

(3) 设置好模拟器的参数后，单击图 1-16 下边的 OK 按钮即可创建一个 AVD。创建好的 AVD 将会显示在如图 1-17 所示的 Android Virtual Device Manager 窗口的文件列表中。

(4) 选中所创建的 AVD 选项，单击右侧的 Start 按钮，弹出如图 1-18 所示的 Launch Options 窗口。

(5) 单击 Launch Options 窗口下的 Launch 按钮即成功启动 AVD，效果如图 1-19 所示。



图 1-17　创建新的 AVD 界面

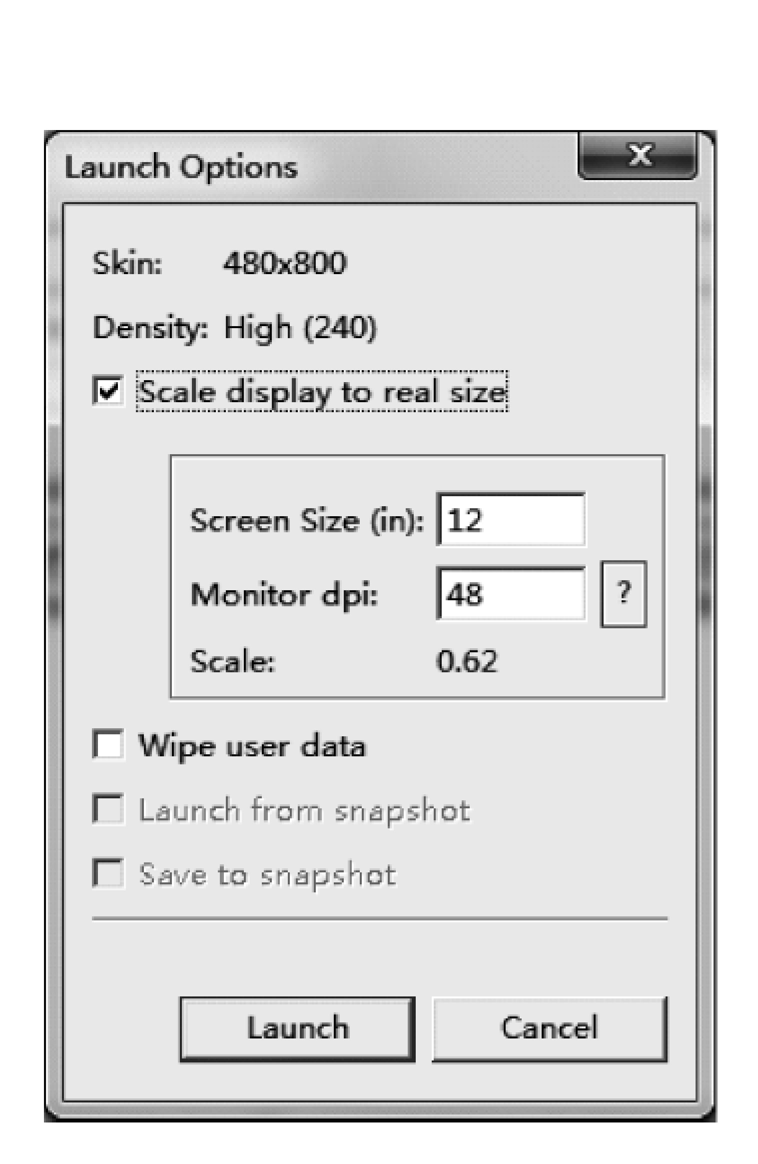


图 1-18　Launch Options 窗口



图 1-19　AVD 界面

使用同样的操作可以根据需要创建多个 AVD 模拟器。这样做的好处是可以模拟程序在不同的 Android 版本上运行的兼容性。

图 1-19 右侧的各个控制按钮名称及其功能如表 1-2 所示。

表 1-2 AVD 的控制按钮功能

模拟器 AVD 的模拟按键	相应的图标	功　　能
音量渐小按钮		控制音量大小
电源按钮		设置电话模式，AVD 开关
问题是增加按钮		控制音量大小
上/下/左/右按钮		确定按钮
中心按钮		上/下/左/右移动焦点
Home 按钮		返回主界面
Menu 按钮		打开应用程序菜单
查询按钮		在手机内部或上网查询
返回按钮		返回上一级界面

1.3　Android 应用组成

Android 的应用项目主要由以下部分组成。

- src 文件：项目源文件都保存在这个目录中。
- R. java 文件：这个文件是 Eclipse 自动生成的，应用开发者不需要去修改里面的内容。
- Android Library：这个应用运行的是 Android 库。
- assets 目录：里面主要放置多媒体等一些文件。
- res 目录：主要放置应用会用到的资源文件。
- drawable 目录：主要放置应用会用到的图片资源。
- layout 目录：主要放置用到的布局文件。这些布局文件都是 XML 文件。
- value 目录：主要放置字符串(strings. xml)、颜色(colors. xml)、数组(arrays. xml)。
- Androidmanifest. xml：相当于应用的配置文件。在这个文件中，必须声明应用的名称，应用所用到的 Activity、Service 以及 receiver 等。

在 Eclipse 中，一个基本的 Android 项目的目录结构如图 1-20 所示。

1. src 目录

与一般的 Java 项目一样，src 目录下保存的是项目的所有包及源文件(. java)，res 目录下包含了项目中的所有资源。例如，程序图标(drawable)、布局文件(layout)和常量(value)等。不同的是，在 Java 项目中没有 gen 目录，也没有每个 Android 项目都必须设有的 AndroidManfest. xml 文件。

. java 格式文件是在建立项目时自动生成的，这个文件是只读模式，R. java 文件是定义该项目所有资源的索引文件。先来看看 Helloworld 项目的 R. java 文件，代码如下：

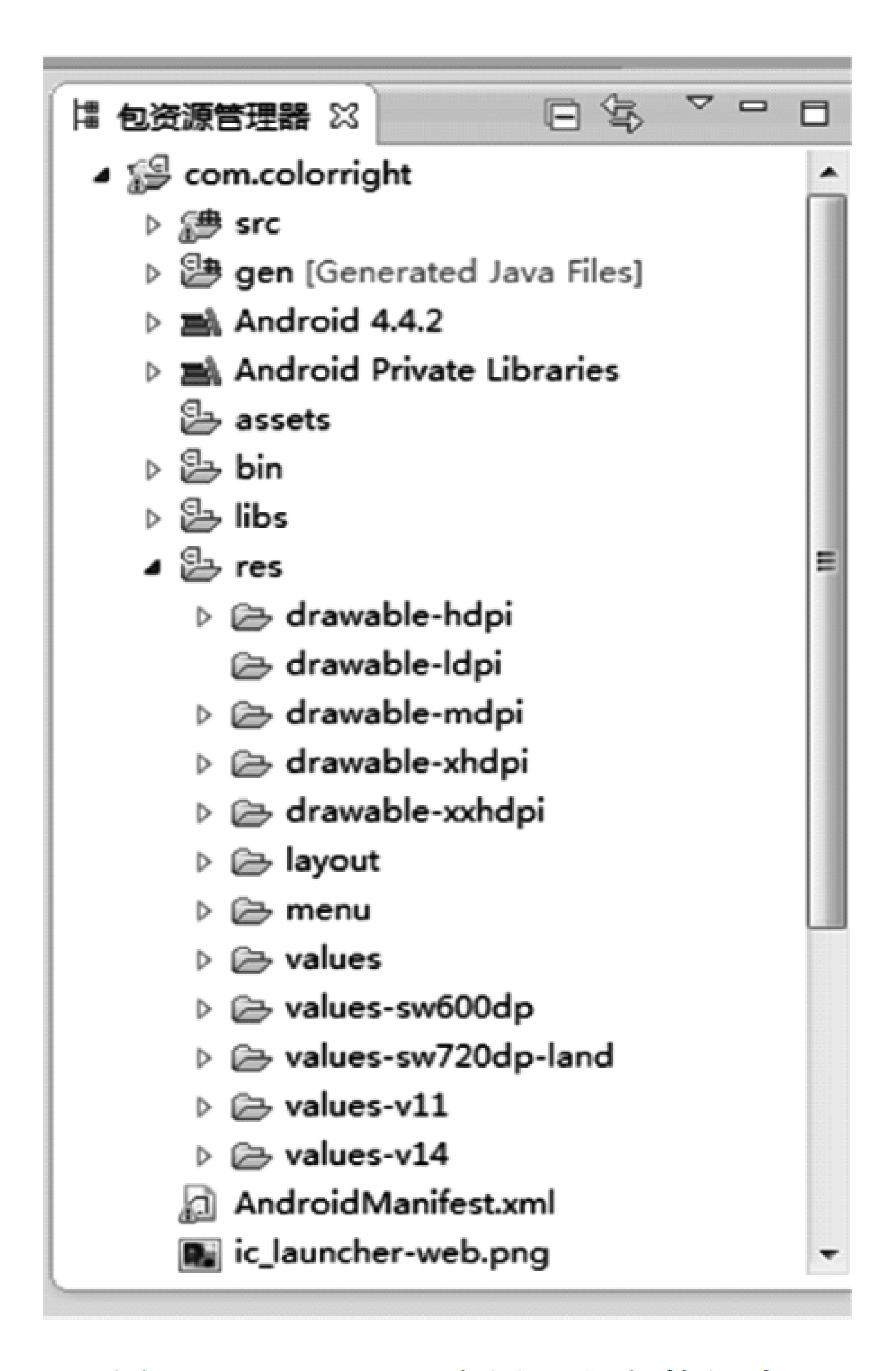


图 1-20　Android 应用工程文件组成

```
package fs.helloworld;
public final class R {
    public static final class attr {
    }
    public static final class dimen {
        public static final int activity_horizontal_margin = 0x7f040000;
        public static final int activity_vertical_margin = 0x7f040001;
    }
    public static final class drawable {
        public static final int ic_launcher = 0x7f020000;
    }
    public static final class id {
        public static final int action_settings = 0x7f080000;
    }
    public static final class layout {
        public static final int main = 0x7f030000;
    }
    public static final class menu {
        public static final int main = 0x7f070000;
    }
    public static final class string {
        public static final int action_settings = 0x7f050001;
        public static final int app_name = 0x7f050000;
        public static final int hello_world = 0x7f050002;
    }
    public static final class style {
        public static final int AppTheme = 0x7f060001;
    }
}
```

从上述代码中，可以看到文件定义了很多常量，并且会发现这些常量的名字都与 res 文件夹中的文件名相同，这再次证明.java 文件中所有存储的都是该项目所有资源的索引。有了这个文件，在程序中使用资源时将变得更加方便，可以很快地找到要使用的资源，由于这个文件不能手动编辑，所以当用户在项目中加入了新的资源时，只需要刷新一下该项目，.java 文件便会自动生成所有资源的索引。

2. res 目录

在 res 目录下包含了该项目所使用到的资源文件，这里面的每一个文件或者资源都将在 R.java 文件中进行索引定义。主要包括如下几类。

- 图片文件：分别提供了高分辨率(drawable-hdpi)、低分辨率(drawable-ldpi)、中分辨率(drawable-mdpi)、超高分辨率(drawable-xhdpi)、超高清分辨率(drawable-xxhdpi)的图片文件。
- 布局文件：在 layout 目录下，默认只有一个 main.xml，用户也可以添加更多的布局文件。
- 字符串：在 values 目录下的 strings.xml 文件中。

打开 main.xml 布局文件，代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity" >
    <TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "@string/hello_world" />
</RelativeLayout>
```

在该布局文件中，首先定义了相对布局，内部只有一个文本框控件。这个控件显示内容引用了 string 文件中的 hello 变量。

其中，

- <RelativeLayout></RelativeLayout>：相对版面配置，在这个标签中，所有元件都是按相对排队排成的。
- android:layout_width：定义当前视图在屏幕上所占的宽度，fill_parent 即填充整个屏幕。
- android:layout_height：随着文字栏位的不同而改变这个视图的宽度或高度。
- android:paddingBottom：指屏幕界面底部的填充方式。
- android:paddingLeft：指屏幕界面左侧的填充方式。
- android:paddingRight：指屏幕界面右侧的填充方式。
- android:paddingTop：指屏幕界面顶部的填充方式。
- tools:context：该布局文件所调用的 Activity 内容。

在上述布局代码中，使用了一个 TextView 来配置文件标签 Widget（构件），其中设置的属性 android：layout_width 为整个屏幕的宽度，android：layout_height 可以根据文字来改变高度，而 android：text 则设置了这个 TextView 要显示的文字内容，这里引用了@string 中的 hello 字符串，即 String. xml 文件中的 hello 所代表的字符串资源。Hello 字符串的内容“HelloWorld、HelloAndroid”这就是用户在 HelloAndroid 项目运行时看到的字符串。

Strings. xml 文件的代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < string name = "app_name"> Hello World </string >
    < string name = "action_settings"> Settings </string >
    < string name = "hello_world"> Hello world! </string >
</resources >
```

3. AndroidManfest. xml 文件

在文件 AndroidManfest. xml 中包含了该项目中所使用的 Activity、Service、Receiver，以下代码为“HelloWorld”项目中的 AndroidManfest. xml 文件。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas.android.com/apk/res/android"   //根节点
    package = "fs.helloworld"                                             //包名
    android:versionCode = "1"
    android:versionName = "1.0" >
    < uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />                                //SDK 版本
    < application                                                         //图标和应用程序名称
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        < activity
            android:name = "fs.helloworld.MainActivity"                   //默认启动的 Activity
            android:label = "@string/app_name" >                          //Activity 名称
            < intent - filter >
                < action android:name = "android.intent.action.MAIN" />
                < category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter >
        </activity >
    </application >
</manifest >
```

1.4 第 1 个 Android 程序

本节将介绍一个简单的 Android 程序的开发过程，让读者对 Android 程序开发流程有一个基本的认识。

1.4.1　Android 开发流程

在创建第 1 个 Android 程序之前，先来了解一下 Android 应用程序的基本开发流程。Android 应用程序的开发流程如下：

(1) 创建 Android 虚拟设备或硬件设备。

开发人员需要创建 Android 虚拟设备(AVD)或链接硬件设备来安装应用程序。

(2) 创建 Android 项目。

Android 项目中包含应用程序使用的全部代码和资源文件。它被构建成可以在 Android 设备安装的.apk 文件。

(3) 构建并运行应用程序。

如果使用 Eclipse 开发工具，每次保存修改时都会自动构建。而且可以单击“运行”按钮来安装应用程序到模拟器。如果使用其他 IDE，开发人员可以使用 Ant 工具进行构建，使用 adb 命令进行安装。

(4) 使用 SDK 调试和日志工具调试应用。

(5) 使用测试框架测试应用程序。

1.4.2　创建应用程序

下面通过一个实例来实现第 1 个 Android 应用程序。其具体实现步骤为：

(1) 启动 Eclipse，进入到 Eclipse 的工作台界面。

(2) 在 Eclipse 的工作台界面中切换新工作空间的具体实现步骤如图 1-21 所示。当单击选择“其他(O)...”选项时，弹出如图 1-22 所示的界面。

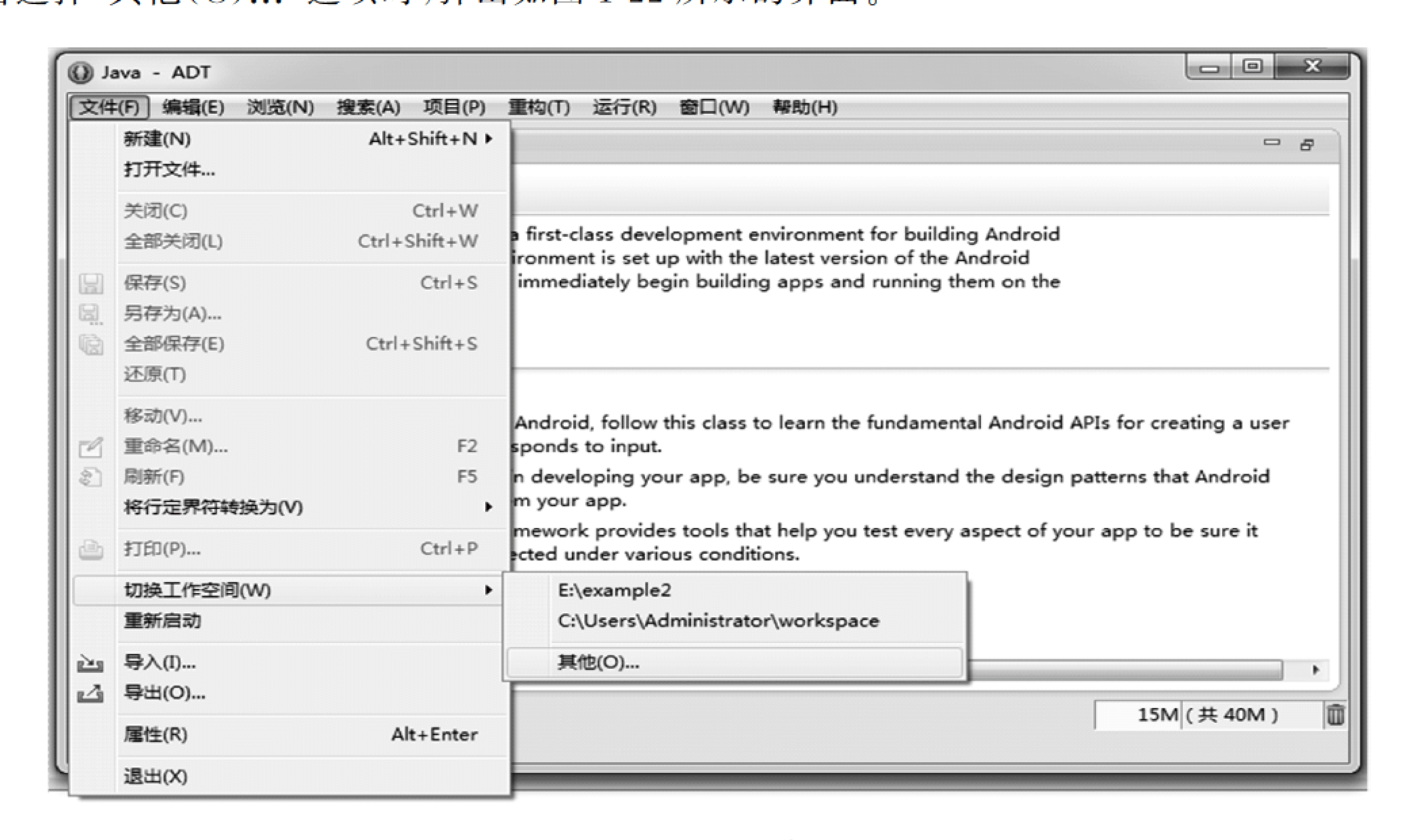


图 1-21　切换工作界面

(3) 在 Eclipse 主界面中，单击“新建”图标 来创建一个 Android 应用项目。Eclipse 弹出如图 1-23 所示的窗口。

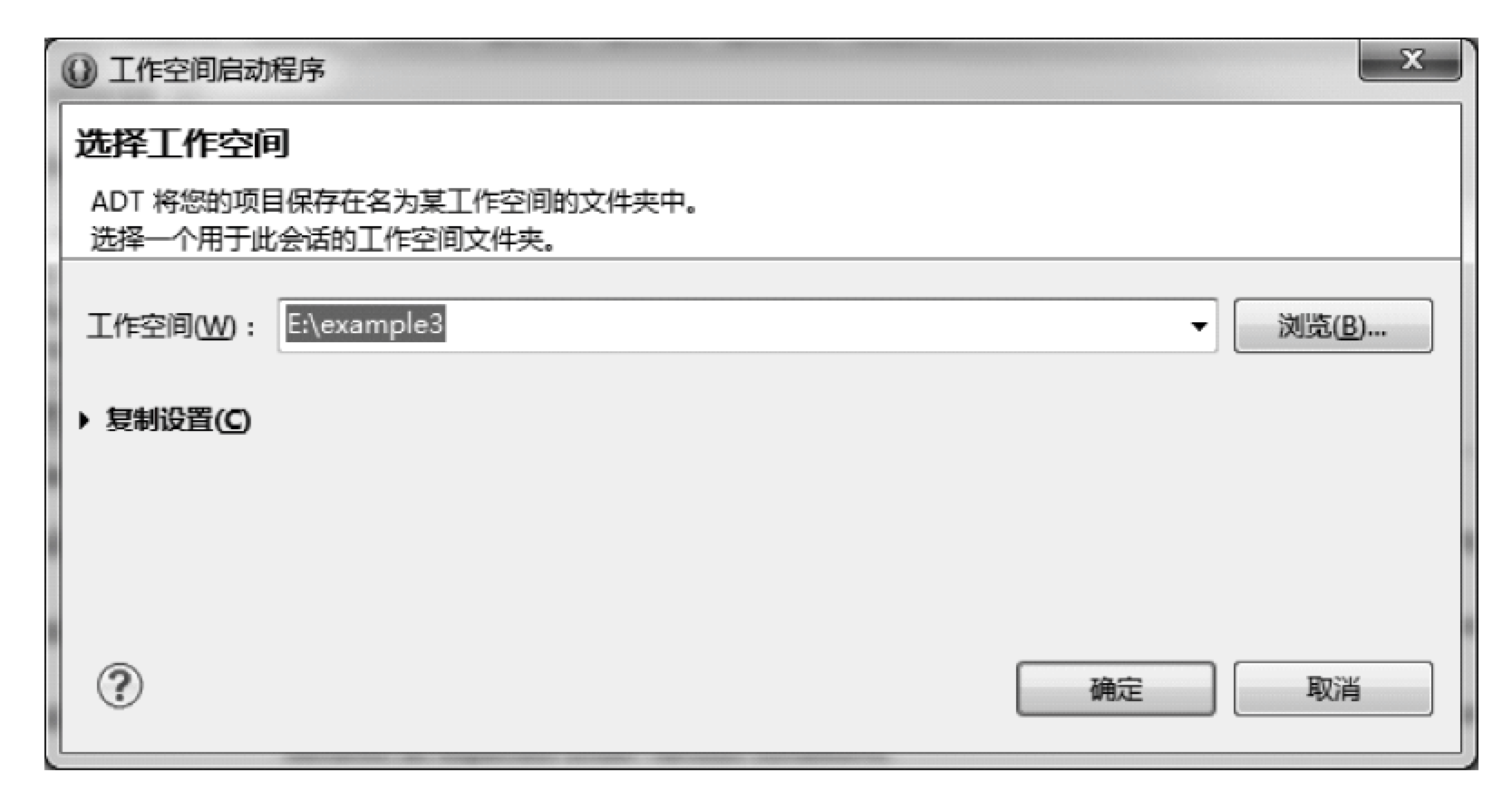


图 1-22　命名新的工作空间

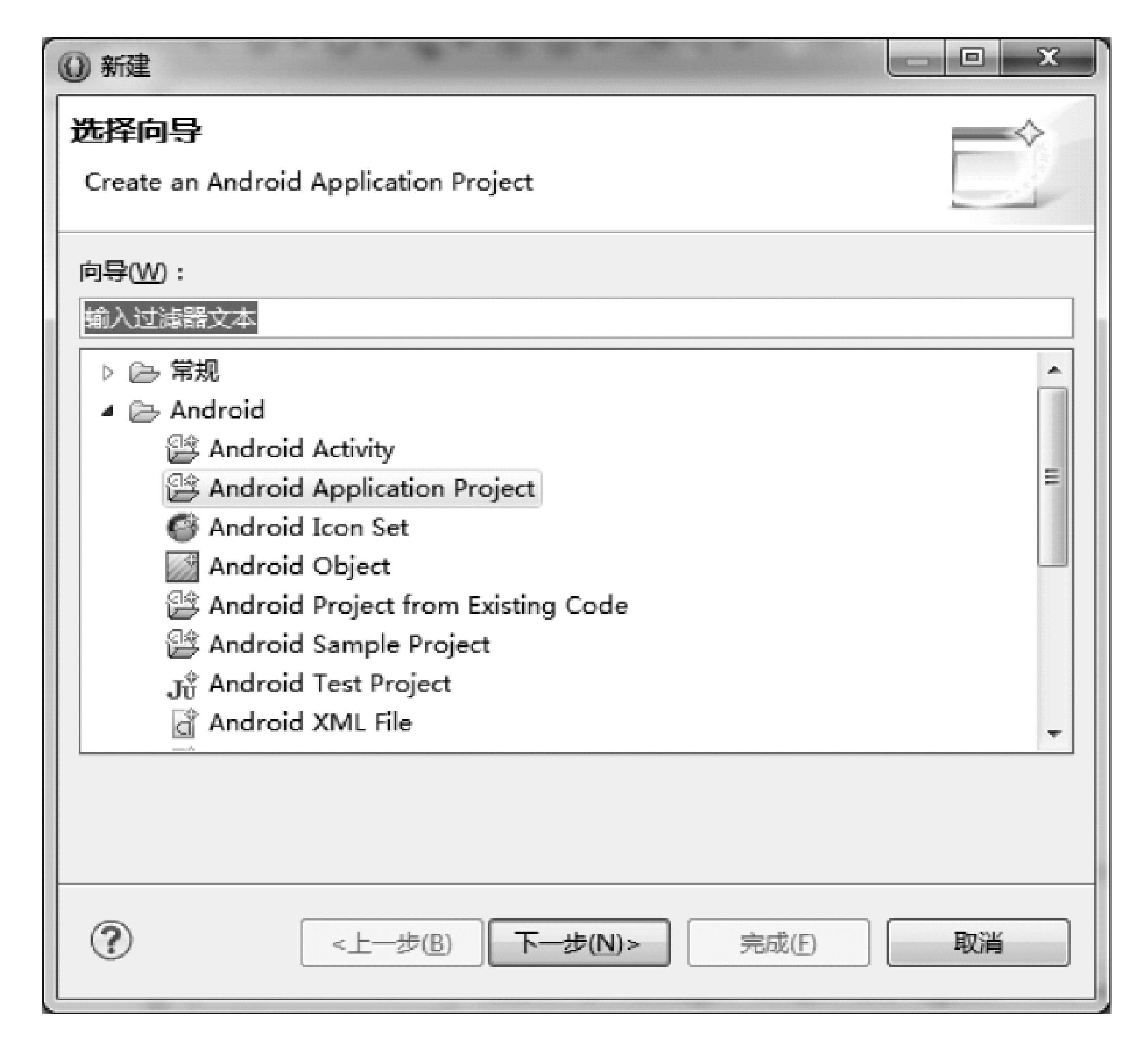


图 1-23　“新建”对话框

(4) 选择图 1-23 中的 Android Application Project 项，之后单击“下一步”按钮，弹出如图 1-24 所示的创建 Android 项目窗口，即为应用项目命名。

应用项目命名填写规则如下。

Application Name：填写应用程序的名称。默认情况下，会将前面填写的项目名称填写在此处，可以进行修改。该名称将作为应用程序的名称出现在手机应用列表中。

Project Name：该栏为工程项目的名称，即在 Eclipse 工作空间创建的文件夹名称，一般以 com.* 形式命名。

Package Name：Java 源文件的包名，Eclipse 会自动在 src 下创建该包名。该包名一般是以 *.* 形式命名。

(5) 单击图 1-24 中的“下一步”按钮，弹出如图 1-25 所示的界面，在该界面中可以确定所创建的 Android 应用项目的内容，如自定义图标、Activity、项目保存的工作空间等。

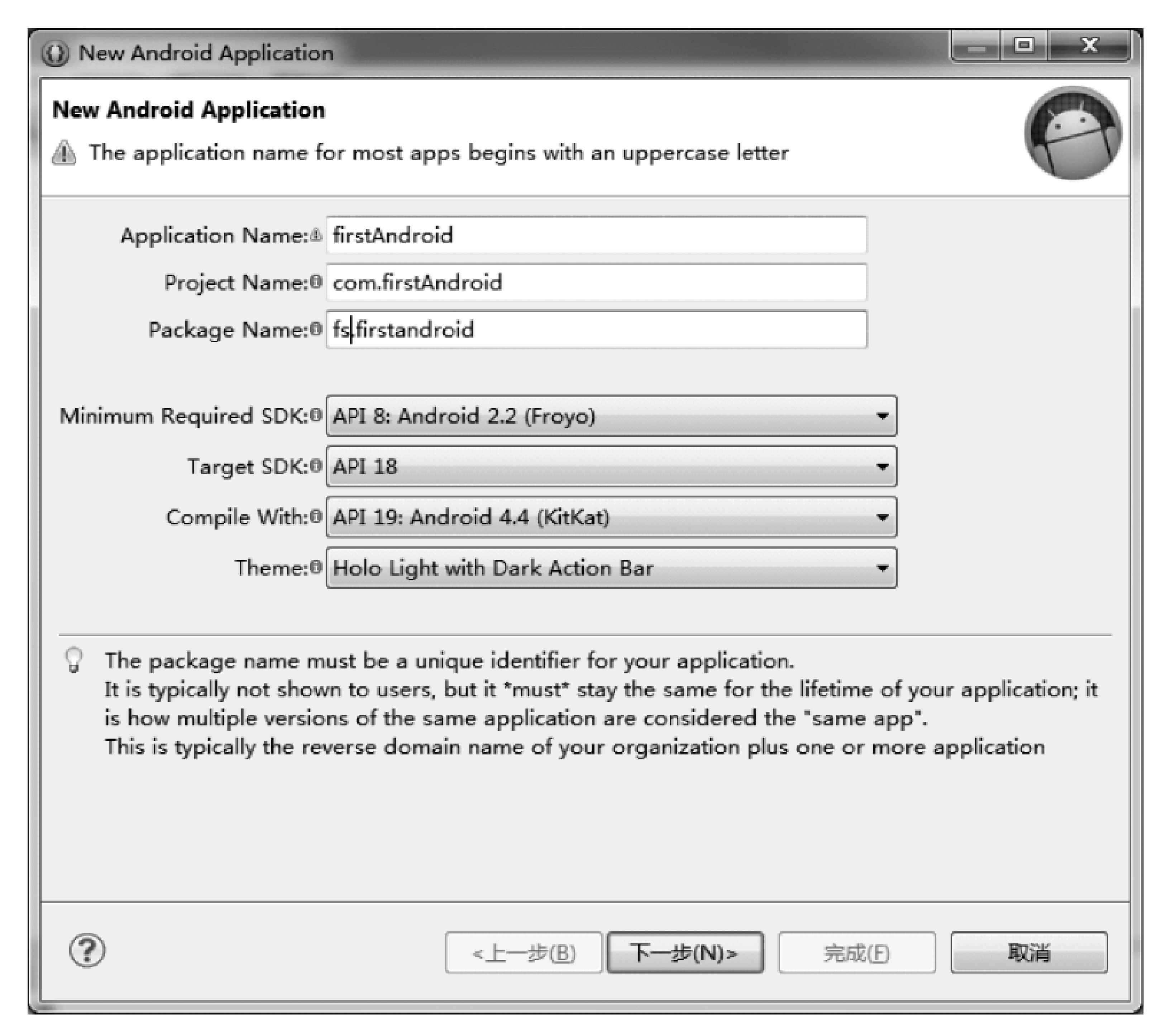


图 1-24　创建 Android 项目窗口

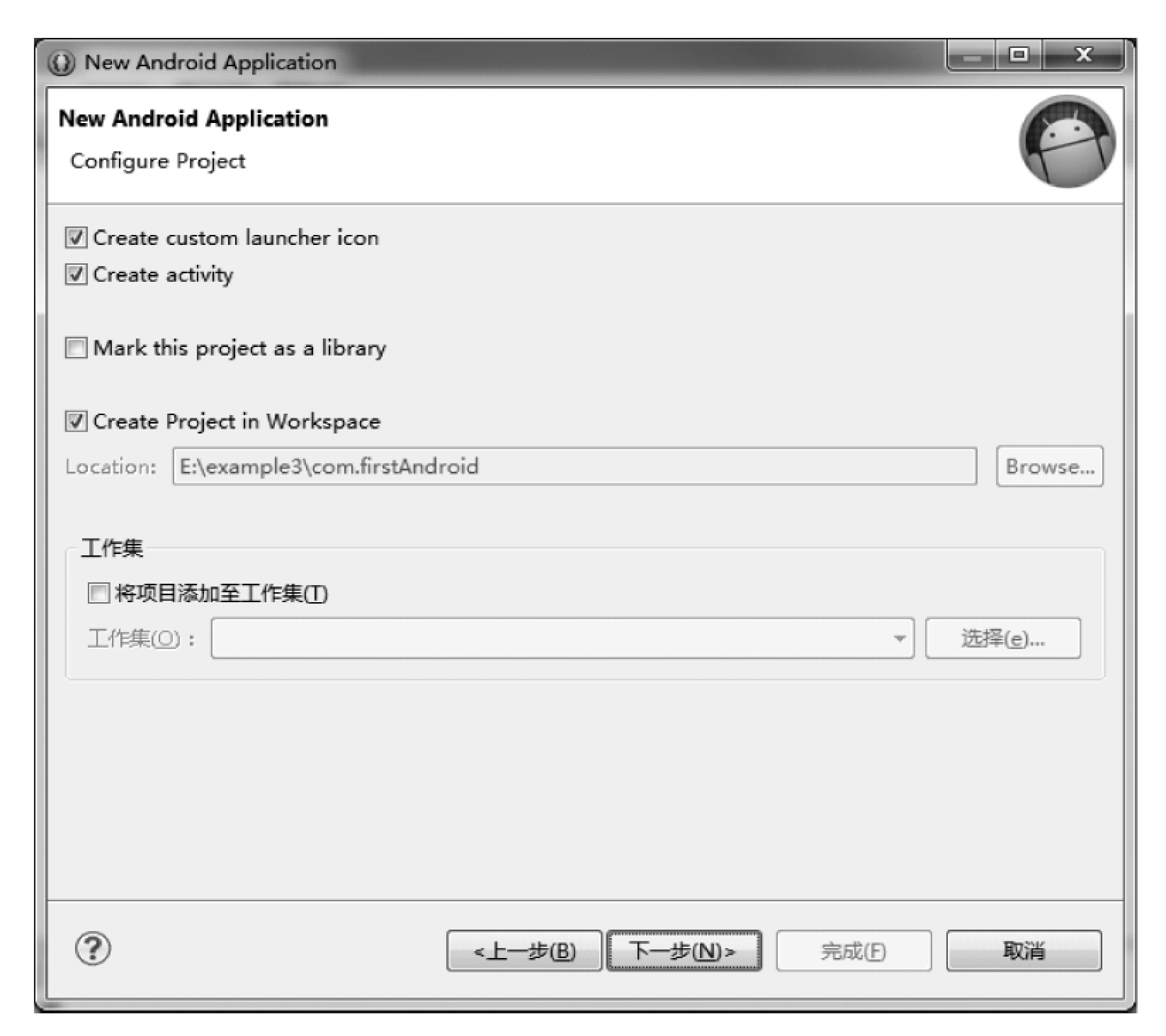


图 1-25　"确定创建应用程序"界面

(6) 单击图 1-25 中的"下一步"按钮，弹出如图 1-26 所示的界面，在该界面中可以自定义应用的图标。

（7）单击图 1-26 中的“下一步”按钮，弹出如图 1-27 所示的界面，在该界面中创建一个空白的 Android 程序。

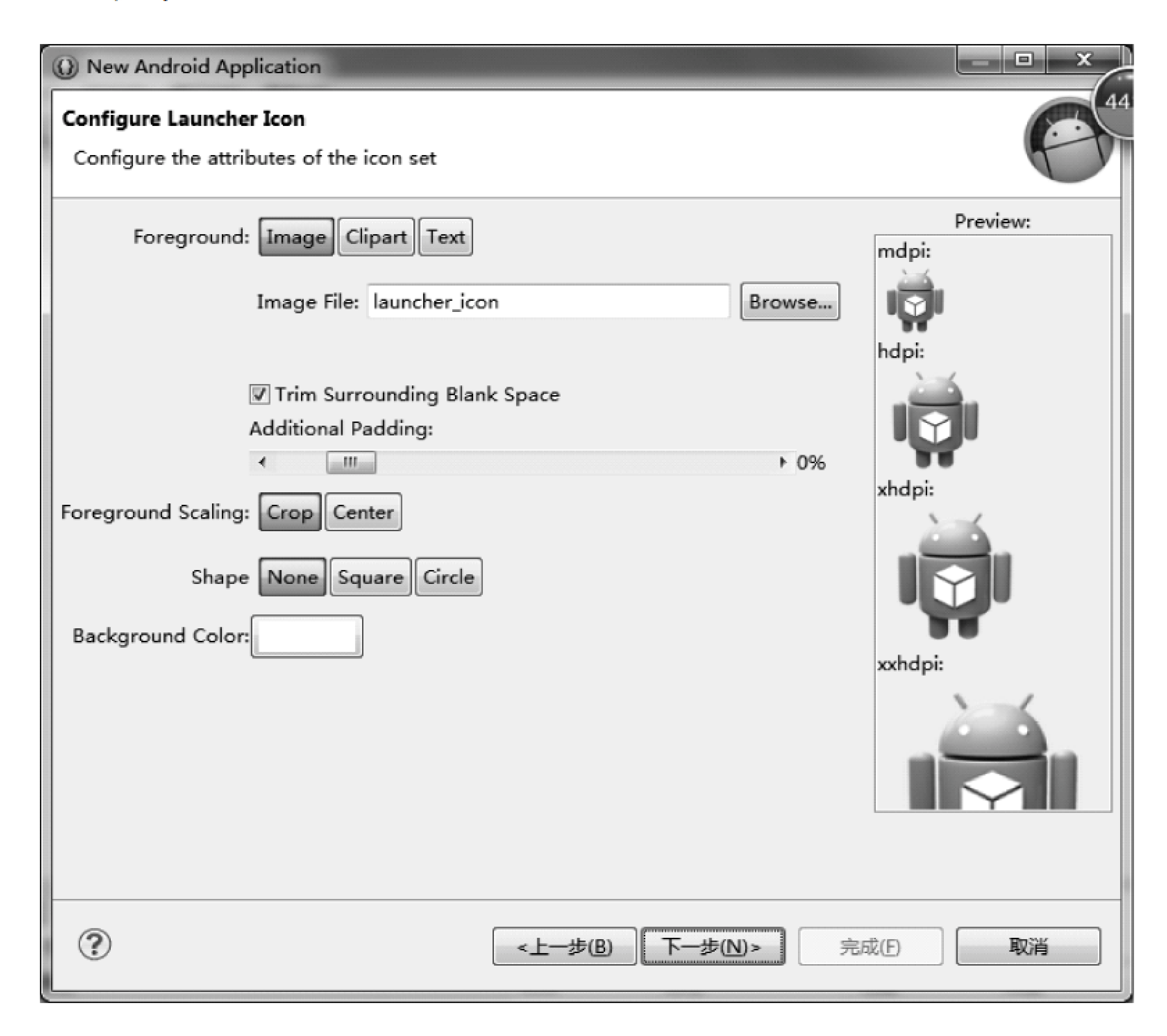


图 1-26　自定义图标界面



图 1-27　空白的 Android 程序

（8）单击图 1-27 界面中的“下一步”按钮，弹出如图 1-28 所示的界面，在该界面中可重新命名主活动名称、布局文件名称以及导航的类型。



图 1-28　重命名界面

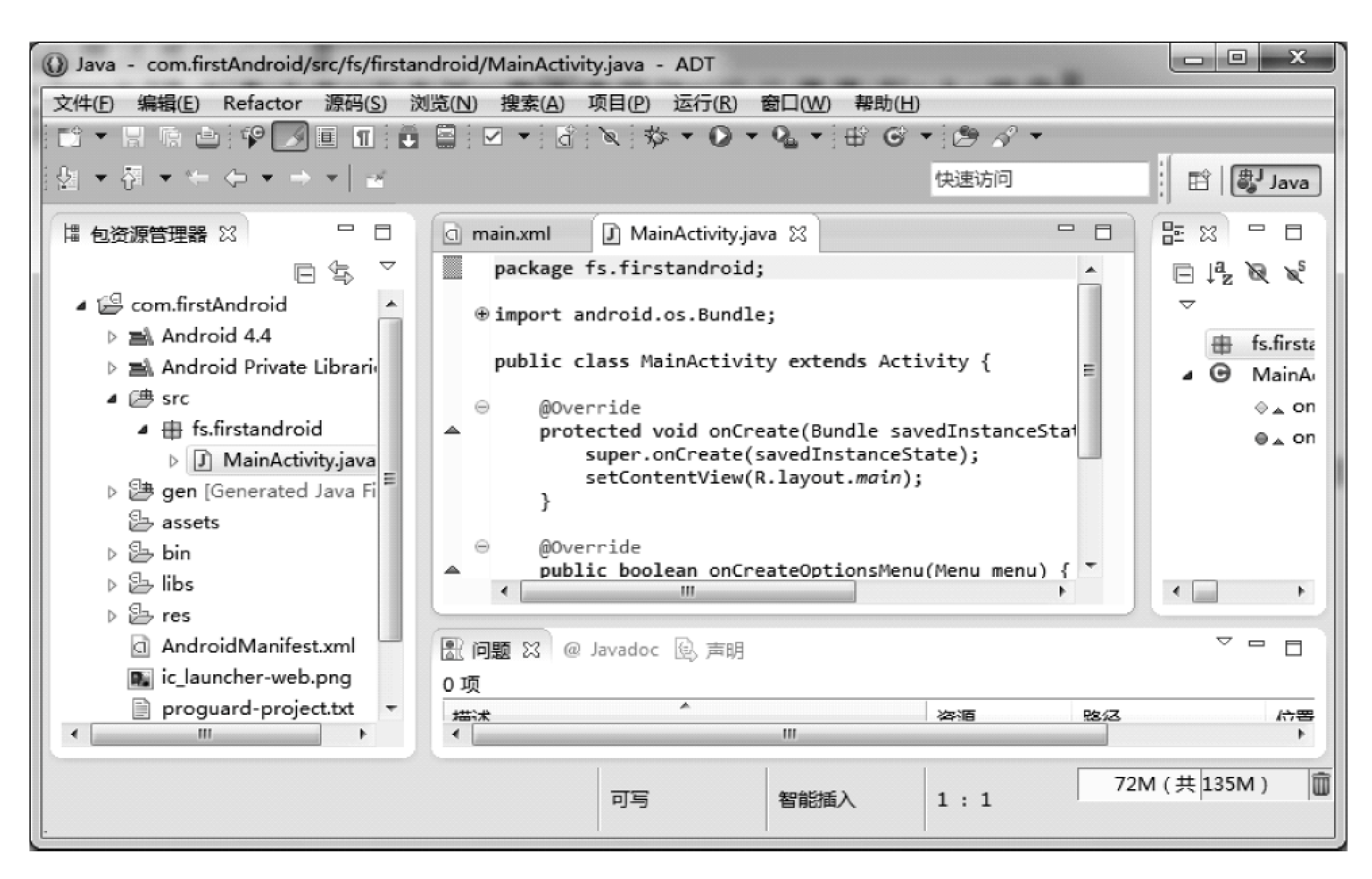


图 1-29　应用项目创建完成界面

(9) 单击图 1-28 界面中的“下一步”按钮，即可完成 Android 应用项目的创建，效果如图 1-29 所示。

图 1-29 中左侧的树形窗口为 HelloAndroid 应用项目的根目录，中间为应用项目的主活动程序，右侧为应用项目的大纲提要。

(10) 默认创建的应用项目 res\layout 目录下的 main. xml 布局文件的默认代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity" >
    <TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "@string/hello_world" />
</RelativeLayout>
```

在文件中定义一个相对布局 RelativeLayout，并在布局文件中声明一个 TextView 控件。

应用项目的 src\fs. helloandroid 包下的 MainActivity. java 是主活动文件，默认代码为：

```
package fs.helloandroid;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        //Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main,menu);
        return true;
    }
}
```

(11) 单击图 1-29 中的“运行”按钮图标，或选中程序后右击，在弹出的快捷菜单中选择“运行方式(R)/Android Application”选项，即可弹出如图 1-30 所示的运行方式。

(12) 在图 1-30 中选择 Android Application 选项，之后单击下方的“确定”按钮，即可运行 com. HelloAndroid 项目，效果如图 1-31 所示。




图 1-30　选择运行方式

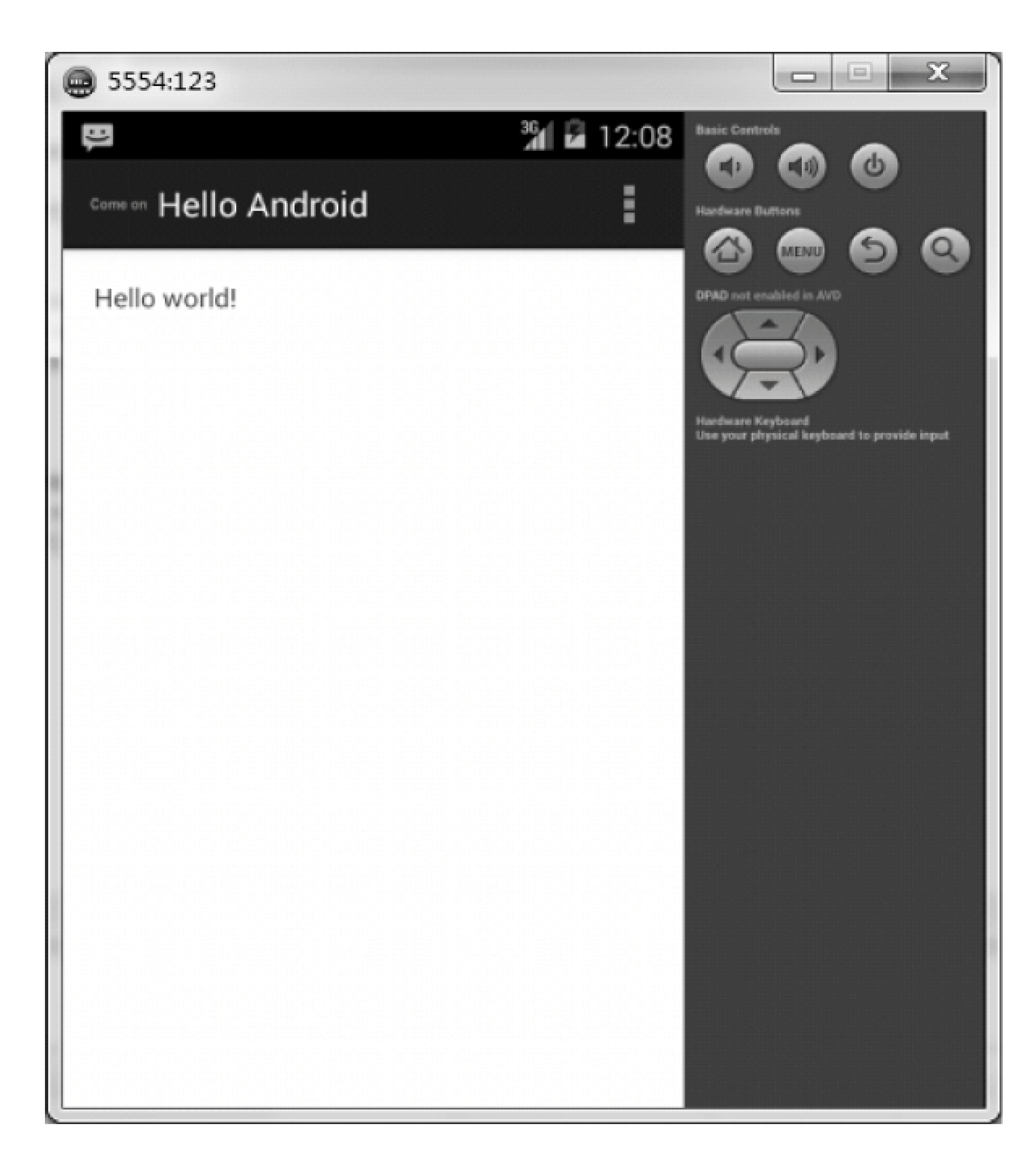


图 1-31　显示界面

1.5　DDMS 使用

在 Android SDK 工具中，提供了 DDMS(Dalvik Debug Monitor Service)来对 Android 的应用程序进行调试和模拟服务，主要提供了针对特定的进程查看正在运行的线程以及堆信息、输入日志(Logcat)、广播状态信息、模拟电话呼叫、接收 SMS、虚拟地理坐标、为测试设备截屏等。

DDMS 会搭建 Eclipse 本地与测试终端(Emulator 或者真实设备)的链接，它们应用各自建立的端口监听调试器的信息，DDMS 可以实时监测到测试终端的连接情况。当有新的测试终端连接后，DDMS 将捕捉到终端的 ID，并通过 adb 工具建立调试器，从而实现发送指令到测试终端的目的。

1. 开启 DDMS 视图

在 Eclipse 的右上角有个 DDMS 图标，单击该图标即可打开 Eclipse 中所有的视图界面。除此之外还可以在 Eclipse 的菜单栏中选择"窗口/打开透视图"选项，在弹出的菜单中选择"其他"选项，效果如图 1-32 所示，即可将打开所有视图。选择图 1-32 中的 DDMS，切换到 DDMS 界面。

2. DDMS 功能

在 DDMS 视图界面中，有调试 Android 设备经常使用的工具，主要包括设备(Devices)、模拟器控制台(Emulator Control)、日志输出(LogCat)、文件浏览器(File Explorer)以及线程、堆栈等。这些功能都显示在 DDMS 界面中。如果在 DDMS 界面中没有找到这些功能

选项，则在 Eclipse 界面菜单栏中选择“窗口/显示视图”选项，选择“其他”选项，将会出现 Eclipse 中所有的功能视图，如图 1-33 所示，选择需要的功能视图进行说明。



图 1-32　打开透视图

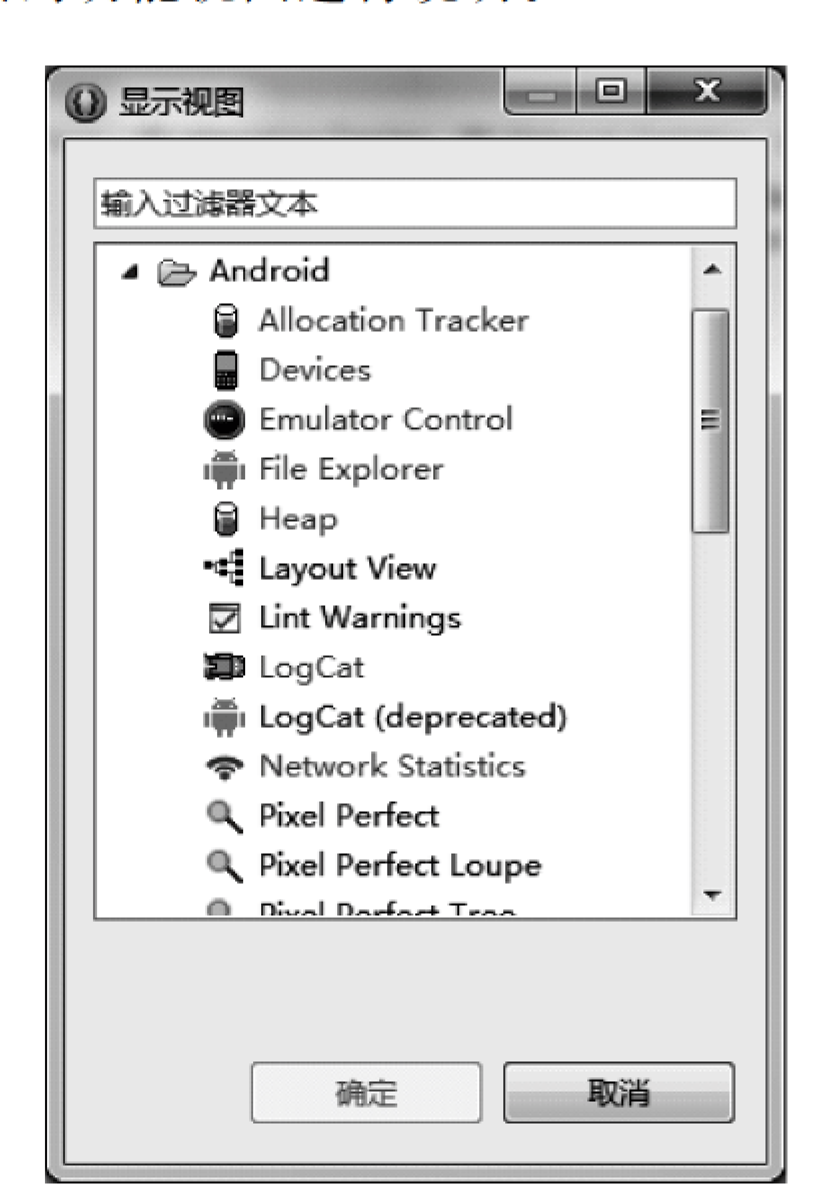


图 1-33　功能视图

在 DDMS 提供的功能中，最常用的主要有以下 4 个。

① 设备（Devices）：设置功能视图一般在 DDMS 的左上角，其标签为 Devices，如图 1-34 所示。在该视图中显示所有连接的 Android 设备并且详细列出该 Android 设备中可连接调试的应用程序进程。从该图中可以看出在列表中从左到右分别是应用程序名和调试器链接的端口号。在进行调试时，一般只需要关心应用程序名。

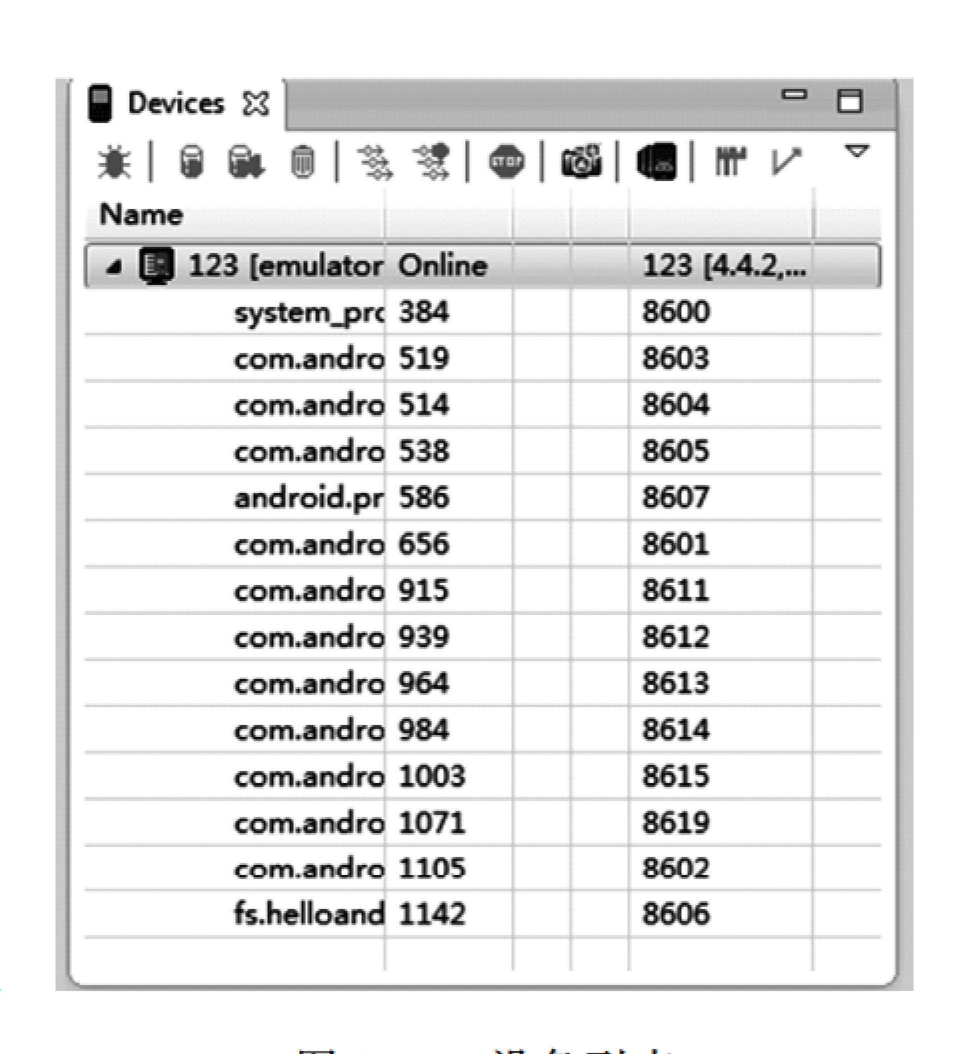


图 1-34　设备列表

当选择了列表中的某一个应用程序时，在视图的右上角有一排功能按钮即可使用。它们主要用于调试某个应用，主要的功能有调试选项（Debug the selected process）、线程查看（Update Threads）、堆栈查看（Update Heap）、终止进程（Stop Process）和截屏（ScreenShot）。

- Debug the selected process：用于显示被选择进程与调试器连接状态。如果进程前带有绿色表示该进程的源文件在 Eclipse 中处于打开状态，并已经开启了调试器监听进程运行情况。
- Update Threads：用于查看当前进程所包含的线程。当选中任意进程后，单击该按钮后，被选中的进程名称后边会出现显示线程信息标识并可以在 Threads 功能界面中看到详细的线程运行情况。
- Update Heap：用于查看当前进程堆栈内存的使用情况。当选中任意进程后，单击

该按钮，可在 Heap 功能界面中看到详细的堆栈使用情况，与 Update Threads 类似。

- Stop Process：终止当前进程。选择进程后，单击该按钮便强制终止了该进程。
- ScreenShot：截取当前测试终端桌面。

② 模拟器控制台(Emulator Control)：由于在模拟器中不断直接使用真机的电话、短信、GPS 位置等功能，当使用模拟区测试这些功能时，可以通过该控制台来实现对这些交互功能的模拟。模拟器控制台视图如图 1-35 所示。

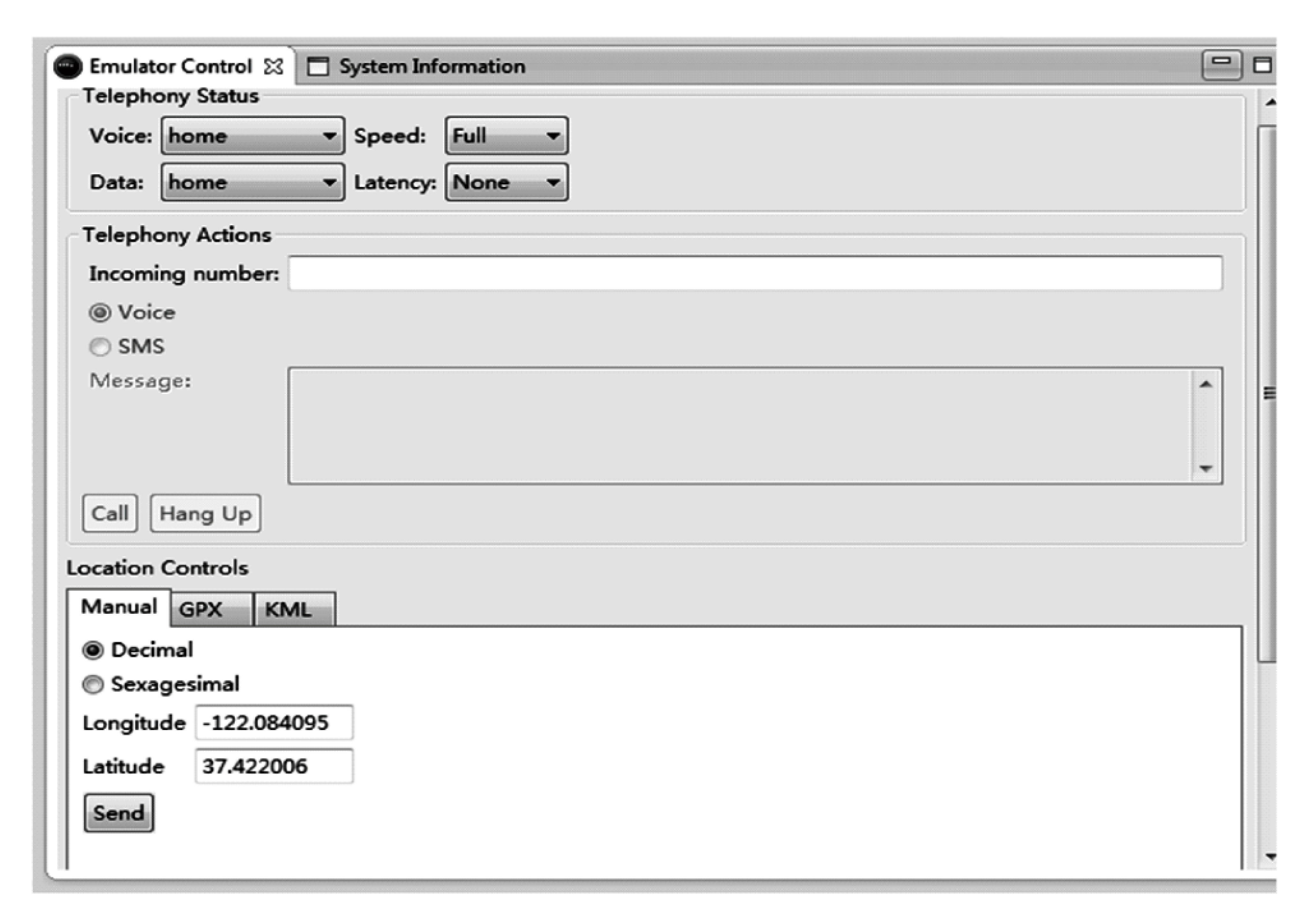


图 1-35　控制台

其各选项说明如下。

- Telephony Status：选择模拟语音质量以及信号连接模式。
- Telephony Actions：模拟电话呼入和发送短信到测试的模拟器。其中 Incoming number 为设置本地呼叫模拟器的号码；Voice 选项表示模拟电话呼入模拟器；SMS 选项表示模拟短信发送到模拟器中。
- Location Controls：模拟地理坐标或模拟动态的路线坐标变化并显示预设的地理标识。其中，有 3 个选项卡表示可以使用不同的 3 种方式，即 Manually 方式，手动为终端发送二维经纬坐标；GPX 方式，通过 GPX 文件导入序列动态变化地理坐标，从而模拟行 GPS 变化的数值；KML 方式，通过 KML 文件导入独特的地理标识，并以动态形式根据变化的地理坐标显示在测试终端。

③ 文件浏览器(File Explorer)：在 DDMS 界面的右边，占用较大一块区域的便是模拟器运行的详细信息，有多个选项卡，其中 File Explorer 即为文件浏览器，如图 1-36 所示。

在文件浏览器中显示 Android 设备的文件系统信息。一般情况下，File Explorer 会有如下 3 个目录：data、mnt 和 system。

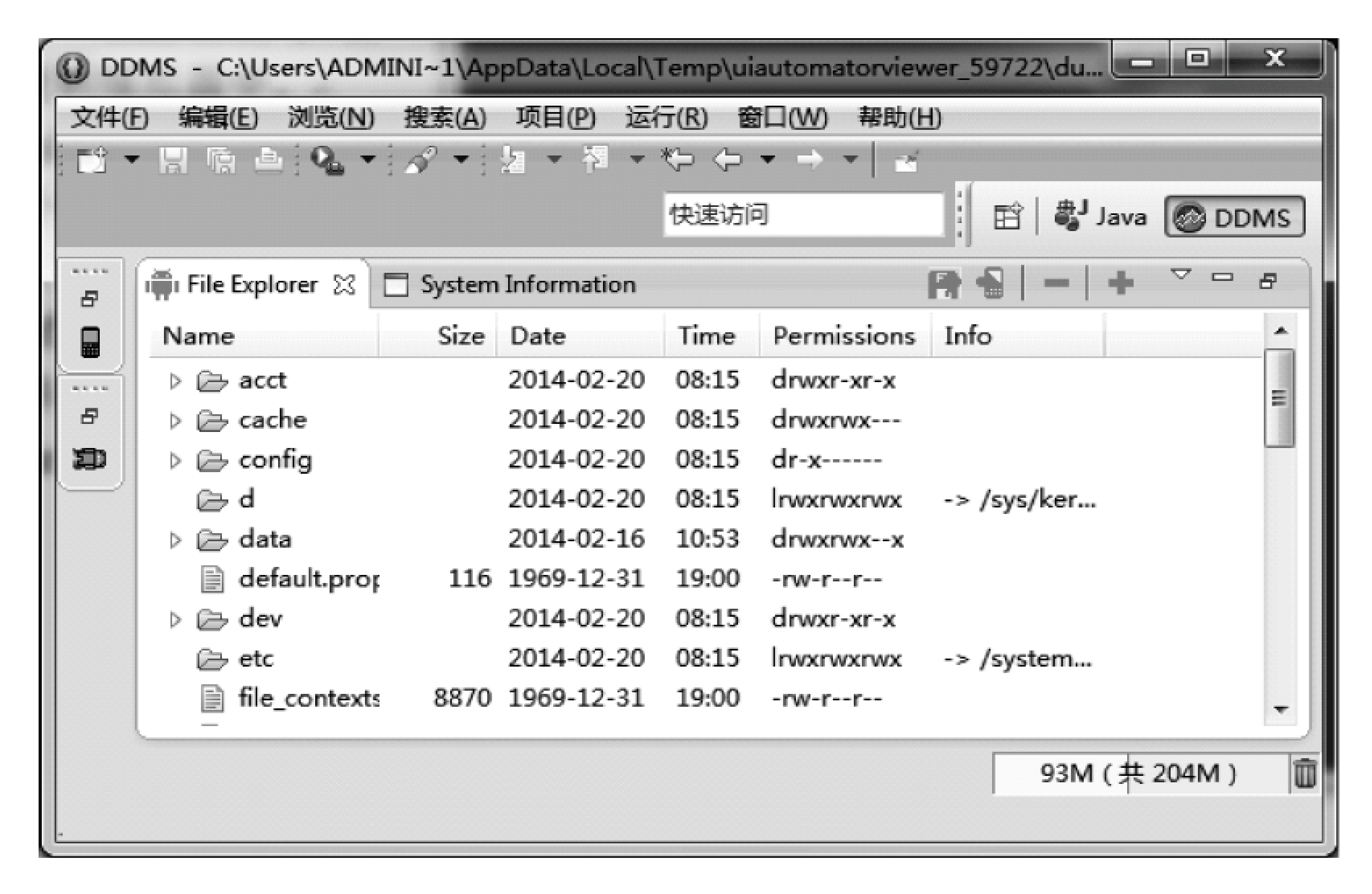


图 1-36　文件浏览器

- data 目录对应手机的 RAM，会存放 Android 系统运行时的 Cache 等临时数据。如果没有 roots 权限，apk 程序将会安装在/data/app 中（只是存放 apk 文件本身）；在/data/data 中存放着所有程序（系统应用程序和第三方应用程序）的详细数据目录信息。
- 在 mnt 目录中最重要的是其目录下的 sdcard 目录。该目录即对应于 SD Card 的目录文件。
- 在 system 目录中对应手机的 ROM，存放 Android 系统以及系统自带的应用程序等。

除了可以查看到这 3 个目录外，还可以使用 File Explorer 对文件进行操作。选项卡右上角的操作按钮从左到右分别为 Android 设备保存到本地、上传到 Android 设备、删除文件、添加文件夹。在使用这 4 个功能时，需要具有对 Android 设备的文件系统相应的操作权限。

④ 日志输出（LogCat）：在模拟器中的所有输出的信息都显示在日志信息中，该视图一般在最下方，如图 1-37 所示。

Javadoc　声明　控制台　LogCat　调试

Search for messages. Accepts Java regexes. Prefix with pid:, app:, ta　verbose

L...	Time	PID	TID	Application	Tag
D	02-20 10:45:01.111	384	423	system_process	Connectivity...
D	02-20 10:45:01.111	384	423	system_process	Connectivity...
D	02-20 10:45:42.451	384	423	system_process	MobileDataSt...
I	02-20 10:48:00.191	384	399	system_process	ProcessStats...
W	02-20 10:49:12.311	384	399	system_process	ProcessCpuTr...

图 1-37　日志信息

在 LogCat 中显示所有测试终端操作的日志记录，通过不同颜色的输出可以很明显地区分警告信息和错误信息，并且可以使用右边的下拉菜单进行不同类型信息的筛选。

1.6 Android 模拟器

Android 模拟器是一个基于 QEMU 的程序，它提供了可以运行 Android 应用的虚拟 ARM 移动设备。它在内核级别中运行一个完整的 Android 系统栈，其中包含了一组可以在自定义应用中访问的预定义应用程序(例如拨号器)。开发人员既可以通过定义 AVD 来选择模拟器运行的 Android 系统版本，还可以自定义移动设备皮肤和键盘映射。在启动和运行模拟器时，开发人员可以使用多种命令和选项来控制模拟器行为。

随 SDK 分发的 Android 系统镜像包含了用于 Android Linux 内核的 ARM 机器码、本地库、Dalvik 虚拟机和不同的 Android 包文件(例如 Android 框架和预定义安装应用)。模拟器 QEMU 层提供了从 ARM 机器码到开发者系统和处理器架构的动态二进制翻译。

通过向底层 QEMU 服务增加自定义功能，Android 模拟器支持多种移动设备的硬件特性，例如：

- ARMv5 中央处理器和对应的内存管理单元(MMU)。
- 16 位液晶显示器。
- 一个或多个键盘(基于 Qwerty 键盘和相关的 Dpad/Phone 键)。
- 具有输出和输入能力的声卡芯片。
- 闪存分区(通过计算机上的硬盘镜像文件模拟)。
- 包括模拟 SIM 卡的 GSM 调制解调器。

1.6.1 Android 虚拟设备

Android 虚拟设备(AVD)是模拟器的一种配置。开发人员通过定义需要硬件和软件选项来使用 Android 模拟器模拟真实的设备。

一个 Android 虚拟设备(AVD)由以下几部分组成。

1. 硬件配置

定义虚拟设备的硬件特性。例如，开发人员可以定义该设备是否包含摄像头、是否使用物理 QWERTY 键盘和拨号键盘、内存大小等。

2. 映射的系统镜像

开发人员可以定义虚拟设备运行的 Android 平台版本。

3. 其他选项

开发人员可以指定需要使用的模拟器皮肤，这将控制屏幕尺寸、外观等。此外，还可以指定 Android 虚拟设备使用的 SD 卡。

4. 开发计算机上的专用存储区域

用于存储当前设备的用户数据(安装的应用程序、设置等)和模拟 SD 卡。

根据需要模拟设备的类型不同，开发人员可以创建多个 AVD。由于一个 Android 应用通常可以在很多类型的硬件设备上运行，开发人员需要创建多个 AVD 进行测试。

为 AVD 选择系统镜像目标时，请牢记以下要点：

(1) 目标的 API 等级非常重要。在应用程序的配置文件(AndroidManifest 文件)中，使用 minSdkVersion 属性标明了需要使用的 API 等级。如果系统镜像等级低于该值，将不能

运行这个应用。

(2) 建议开发人员创建一个 API 等级大于应用程序所需等级的 AVD，这主要用于测试程序的向后兼容性。

(3) 如果应用程序配置文件中说明需要使用额外的类库，则只能在包含该类库的系统镜像中运行。

1.6.2 Android 模拟器限制

在当前的版本中，模拟器有如下限制：

- 不支持拨打或接听真实电话，但是可以使用模拟器控制台模拟电话呼叫。
- 不支持 USB 连接。
- 不支持相机/视频采集(输入)。
- 不支持设备连接耳机。
- 不支持确定连接状态。
- 不支持确定电量水平和交流充电状态。
- 不支持确定 SD 卡插入/弹出。
- 不支持蓝牙。

1.6.3 Android 模拟器按键

用户可以使用启动选项和控制台命令来控制模拟器环境的行为和特性。当模拟器运行时，用户可以像使用真实移动设备那样使用模拟移动设备。不同的是需要使用鼠标来"触摸"触摸屏，使用键盘来"按下"按键。

表 1-3 列出了模拟器按键与键盘按键对应的关系。

表 1-3 模拟器按键对应的键盘按键

模拟器按键	键 盘 按 键
Back	Esc 键
Call	F3 键
Hangup	F4 键
Home	Home 键
Menu	F2 键或 Page Up 键
Power	F7 键
Search	F5 键
音量减少	KEYPAD_MINUS 或 Ctrl+F6
音量增加	KEYPAD_PLUS 或 Ctrl+F5
切换到先前的布局方向(如横向或纵向)	KEYPAD_7
切换到下一个布局方向(如横向或纵向)	KEYPAD_9
开启/关闭电话网络	F8 键
切换轨迹球模式	F6 键
切换全屏模式	Alt+Enter
透明度/减少	KEYPAD_MULTIPLY(*)/KEYPAD_DIVIDE(/)
临时进入轨迹球模式(当键按下时)	Delete 键

1.7 Android 模拟器操作

在 1.2.5 节已经介绍怎样创建一个 Android 模拟器，本节将介绍如何删除 Android 模拟器以及为 Android 模拟器设置语言、输入法、日期等。

1.7.1 删除模拟器

删除 Android 模拟器的步骤比较简单，只需要在 Android Virtual Device Manager 窗口中选中要删除的 Android 模拟器，然后单击 Delete 按钮即可，如图 1-38 所示。

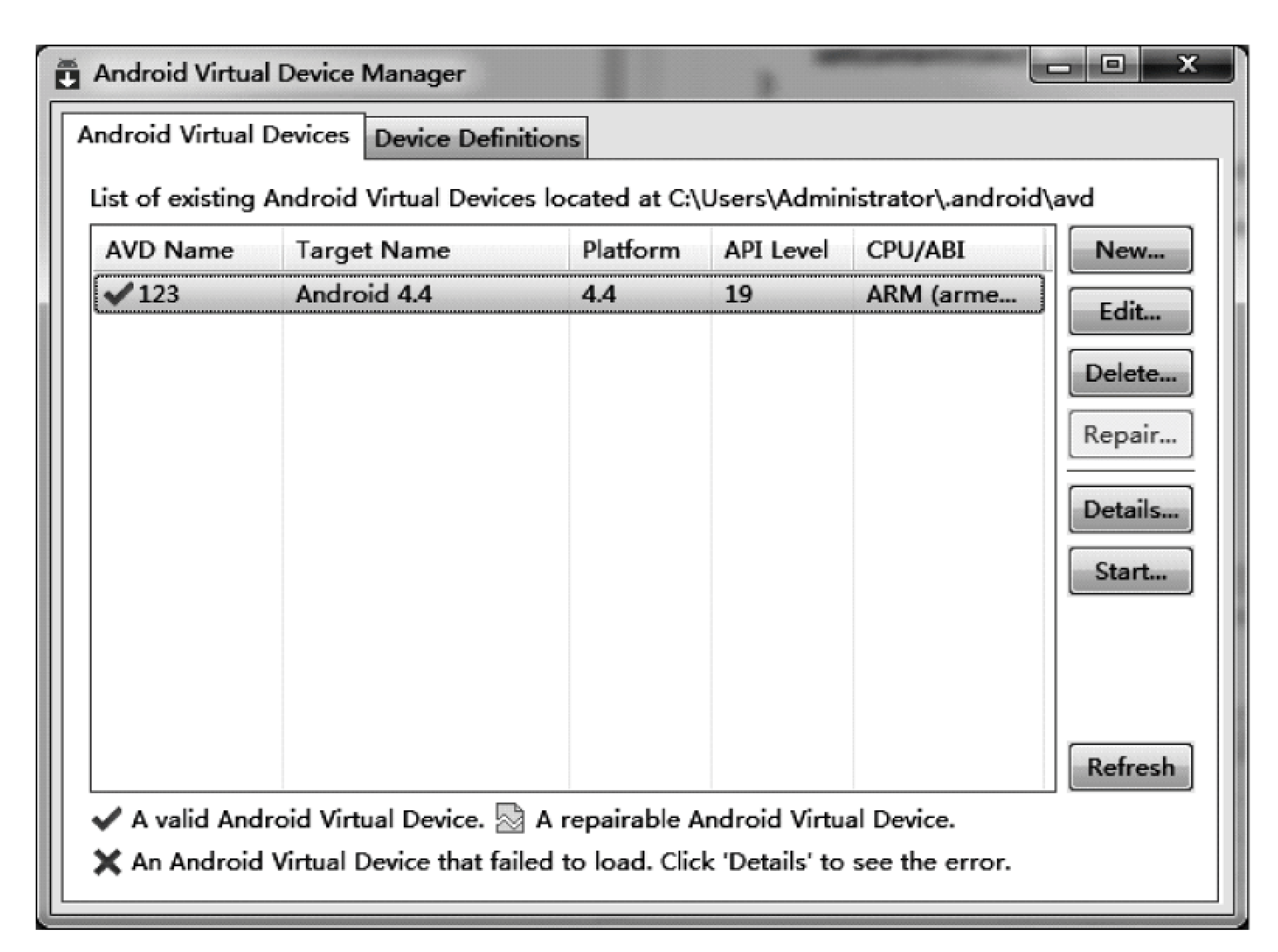


图 1-38 删除 Android 模拟器

1.7.2 设置语言

Android 模拟器启动后，默认的语言是英语，为了更方便中国区用户的使用，可以将默认语言设置为中文。其具体步骤为：

（1）打开 Android 模拟器并解除锁定，如图 1-39 所示。

（2）单击 Android 主界面最底端的中间按钮，进入 Android 应用程序界面，找到 Setting 按钮（如果在第一页中没有该图标，可以通过左右翻页来进行查找），如图 1-40 所示。

（3）单击 Setting 按钮，进入 Android 模拟器的设置界面，在 Android 模拟器的设置界面中选择 Language & input 选项，如图 1-41 所示。

（4）在打开的列表中选择 Language 选项，如图 1-42 所示。

（5）进入语言选择列表界面，如图 1-43 所示。在列表中找到“中文（简体）”列表项，选中该列表项，这样即可将 Android 模拟器的默认语言设置为中文。

（6）将默认语言设置为中文（简体）后，Android 应用程序主界面的效果如图 1-44 所示。



图 1-39　模拟器主界面

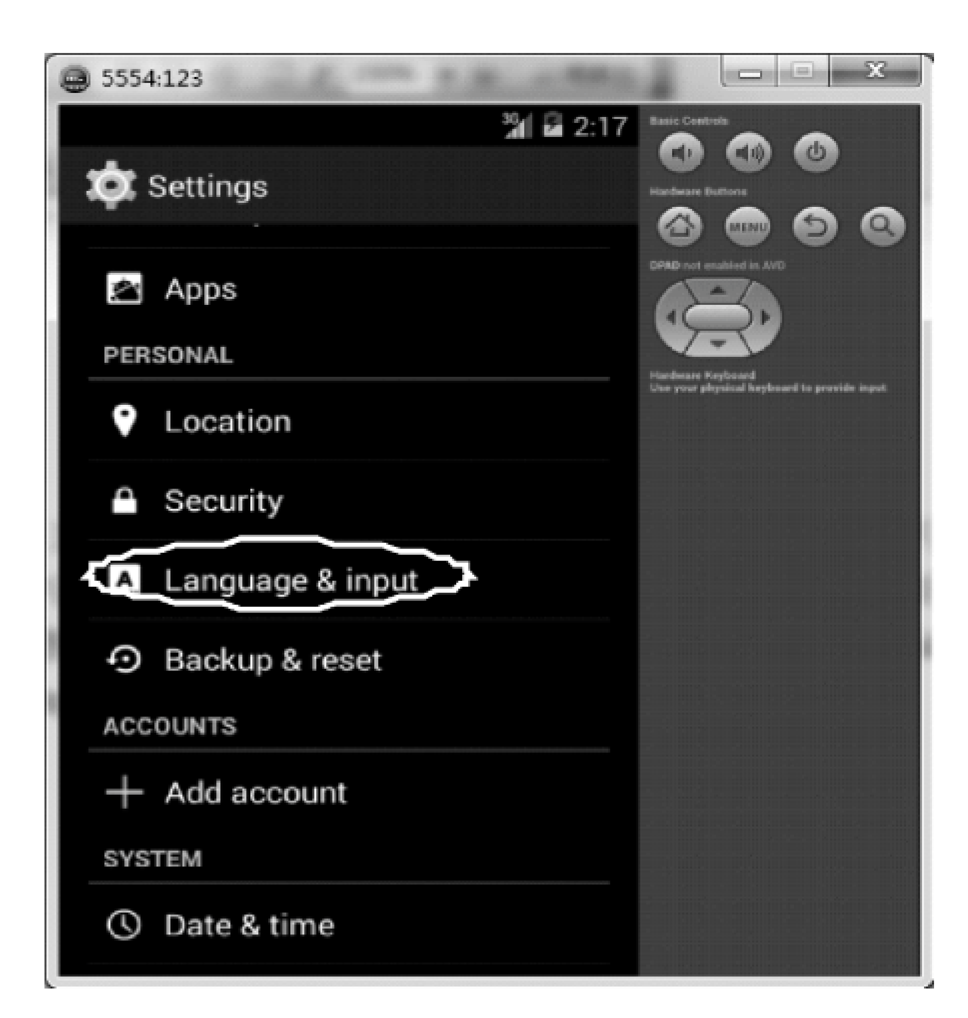


图 1-40　Android 应用界面



图 1-41　选择 Language & input 选项

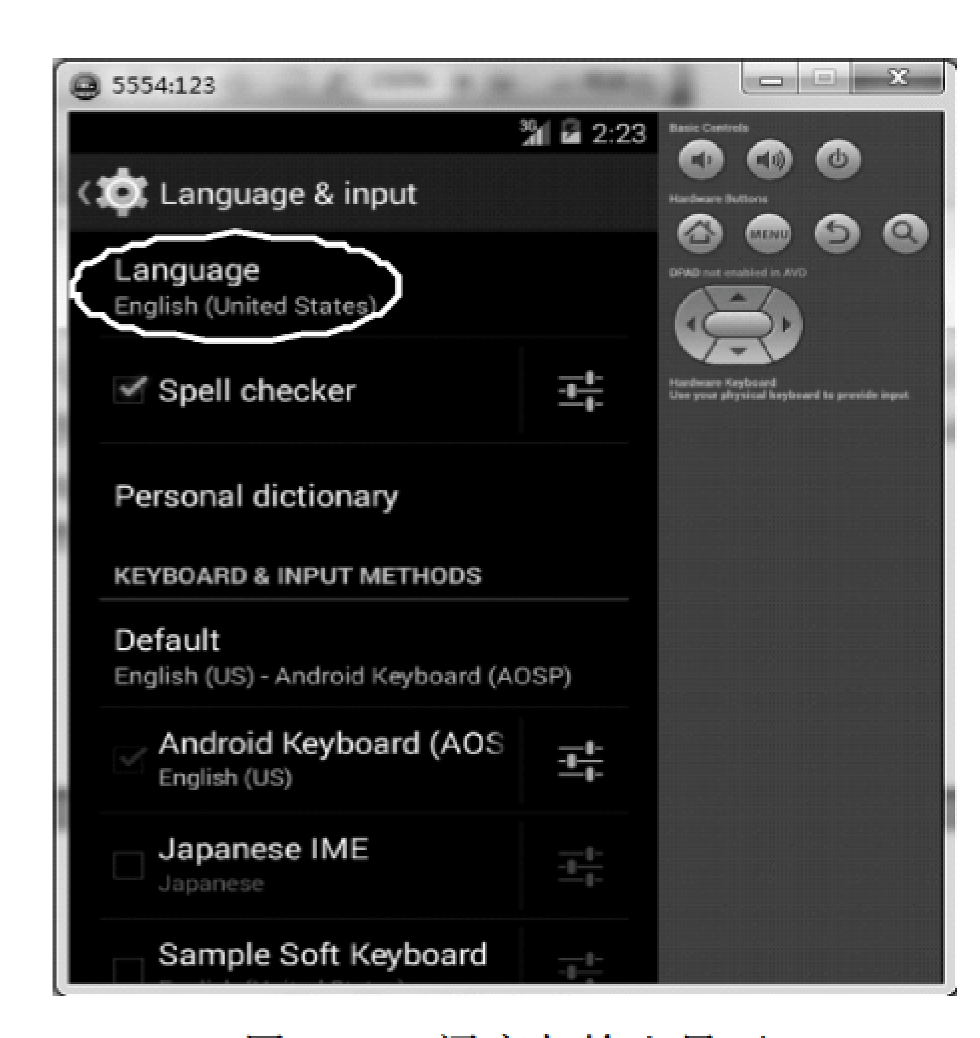


图 1-42　语言与输入界面



图 1-43　语言界面



图 1-44　设置中文后的 Android 应用程序界面

1.7.3 设置输入法

Android 模拟器启动后，默认输入法为 Android 键盘(AOSP)，用户可以根据自己的使用习惯对输入法进行设置，在此介绍怎样在 Android 模拟器中设置输入法。其具体步骤为：

(1) 在 Android 模拟器的设置界面中选择“语言和输入法”选项，如图 1-45 所示。

(2) 在打开的列表中选中“谷歌拼音输入法”复选框，并单击“默认”列表项，如图 1-46 所示。



图 1-45 选择“语言和输入法”选项



图 1-46 单击“默认”列表项

(3) 在打开的如图 1-47 所示的“选择输入法”对话框中列出了 Android 模拟器自带的几种输入法，用户可以根据自己的习惯选择相应的输入法，在此选中“谷歌拼音输入法”单选按钮，这样即可将 Android 模拟器的默认输入法设置为中文输入法。



图 1-47 “选择输入法”对话框

1.7.4 设置日期时间

Android 模拟器启动后，默认时间为格林尼治时间，在此介绍怎样将默认时间设置为中国标准时间。具体实现步骤为：

(1) 打开 Android 模拟器，进入设置界面，选择"日期和时间"列表项，如图 1-48 所示。

(2) 进入"日期和时间"界面，如图 1-49 所示。在该界面中，首先将"自动确定日期和时间"和"自动确定时区"两个复选的选中状态取消掉，之后单击"选择时区"列表项。



图 1-48 选择"日期和时间"列表项



图 1-49 "日期和时间"界面

(3) 进入"日期和时间——选择时区"界面，在该界面中选择"中国标准时间(北京)"列表项，如图 1-50 所示。



图 1-50 "日期和时间——选择时区"界面

(4) 返回如图 1-49 所示的“日期和时间”界面，在该界面中单击“设置日期”列表项，弹出“设置日期”对话框，在该对话框中设置 Android 模拟器的日期，如图 1-51 所示。

(5) 返回如图 1-49 所示的“日期和时间”界面，在该界面中单击“设置时间”列表项，弹出“设置时间”对话框，在该对话框中设置 Android 模拟器的时间，如图 1-52 所示。



图 1-51 “设置日期”对话框



图 1-52 时间设置

(6) 返回如图 1-49 所示的“日期和时间”界面，用户还可以通过单击该界面中的“使用 24 小时格式”和“选择日期格式”列表项，设置 Android 模拟器的日期和时间格式。通过以上步骤，即可完成 Android 模拟器的日期和时间设置。

1.8 应 用

在 Android 的模拟器中，提供了多种桌面背景，下面将介绍怎样进行设置。

(1) 启动模拟器，进入设置列表，在设置列表中找到“显示”列表项，如图 1-53 所示。

(2) 在图 1-53 中单击“显示”列表项，进入显示界面，在该界面中找到“壁纸”列表项，如图 1-54 所示。

(3) 在图 1-54 中单击“壁纸”列表项，显示“选择壁纸来源”界面，如图 1-55 所示。

(4) 在图 1-55 中单击“壁纸”列表项，进入如图 1-56 所示的界面。在该界面中，滑动下方的画廊视图可以切换不同的背景图片，当前居中显示的背景图片会显示预览效果。单击“设置壁纸”按钮完成设置。



图 1-53　Android 模拟器设置界面



图 1-54　Android 模拟器显示设置界面



图 1-55　Android 模拟器壁纸设置界面



图 1-56　Android 模拟器壁纸选择界面

第 2 章　Android 界面设计

用户界面设计是 Android 应用开发的一项重要内容。在进行用户界面设计时，首先需要了解页面中的 UI 元素是怎样呈现给用户的，即怎样控制 UI 界面，Android 系统提供了 3 种控制 UI 界面的方法。

2.1　UI 界面

3 种控制 UI 界面的方法主要有 XML 布局控制 UI 界面、Java 代码控制 UI 界面以及 XML 和 Java 混合控制 UI 布局，下面分别给予介绍。

2.1.1　XML 布局控制 UI 界面

在 Android 中，有一种非常简单、便捷的方法用于控制 UI 界面。该方法就是采用 XML 文件进行界面布局，从而将布局界面的代码和逻辑控制的 Java 代码分享开来，使程序的结构更加清晰、明了。

使用 XML 布局文件控制 UI 界面可分为以下两个步骤。

(1) 在 Android 应用文件的 res\layout 目录下编写 XML 布局文件，可以采用任何 Java 符号命名规则的文件名。创建后，R. java 会自动收录该布局资源。

(2) 在 Activity 中使用以下 Java 代码显示 XML 文件中布局的内容。

```
setContentView(R.layout.main);
```

其中，main 为 XML 布局文件的文件名。

下面通过一个简单的实例来演示怎样使用 XML 布局文件控制 UI 界面。

【例 2-1】 使用 XML 布局文件控制 UI 界面。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 XML_UI_test。

(2) 打开 res\layout 目录下的 main. xml 布局文件，修改布局文件的代码。在该文件中，采用帧布局(FrameLayout)，并声明两个 TextView 控件，第 1 个 TextView 用于显示提示文字，第 2 个 TextView 用于在窗体的正中间位置显示“开始游戏”按钮。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< FrameLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/bj"     >
    < TextView
```

```
            android:id = "@ + id/textView1"
            android:layout_width = "277dp"
            android:layout_height = "wrap_content"
            android:text = "@string/title" />
        < TextView
            android:id = "@ + id/textView2"
            android:layout_width = "153dp"
            android:layout_height = "38dp"
            android:layout_gravity = "center"
            android:text = "@string/start"   />
</FrameLayout >
```

在布局文件 main. xml 中，通过设置布局管理器的 android:background 属性，可以为窗体设置背景图片，该图片放置在 res\drawble-hdpi 等几个文件中的其中一个。通过设置具体组件的 style 属性，可为组件设置样式。使用 android:layout_gravity="center"用于使该组件在帧布局中居中显示。

(3) 选择 res\values 目录下的 strings. xml 文件，在该文件中添加一个用于定义开始按钮内容的常量，名称为 start，内容为“开始游戏”。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < string name = "title">第 1 个 TextView 文件</string >
    < string name = "app_name"> XML 文件控制 UI 界面</string >
    < string name = "action_settings"> Settings </string >
    < string name = "hello_world"> Hello world! </string >
    < string name = "start">开始游戏</string >
</resources >
```

strings. xml 文件用于定义程序中应用的字符串常量。其中，每一个<string>子元素都可以定义一个字符串常量，常量名称由 name 属性指定，常量内容写在起始标记<strings>和结束标记</string>之间。

(4) 在主活动，也就是 MainActivity 中，用以下代码指定活动应用的布局文件。

```
setContentView(R. layout. main);
```

运行程序，效果如图 2-1 所示。

还可以用另外一种方法设置界面的背景颜色，及对变量的定义。其具体操作为：

(1) 打开 res\layout 目录下的 main. xml 文件，代码修改为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< FrameLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = " # aabbcc"      >
    < TextView
        android:id = "@ + id/textView1"
        android:layout_width = "277dp"
        android:layout_height = "wrap_content"
```

```
            android:text = "第 1 个 TextView 文件" />
        < TextView
            android:id = "@ + id/textView2"
            android:layout_width = "153dp"
            android:layout_height = "38dp"
            android:layout_gravity = "center"
            android:text = "游戏开始 ...... "   />
</FrameLayout >
```

以上代码中,android:background="#aabbcc"用于设置界面的背景颜色,通过 android:text 属性可设置控件的内容。

(2) 打开 res\values 目录下的 strings.xml 文件,在该文件中设置界面的标题,代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < string name = "app_name"> XML 文件控制 UI 界面</string >
    < string name = "action_settings"> Settings </string >
    < string name = "hello_world"> Hello world! </string >
</resources >
```

运行程序,效果如图 2-2 所示。

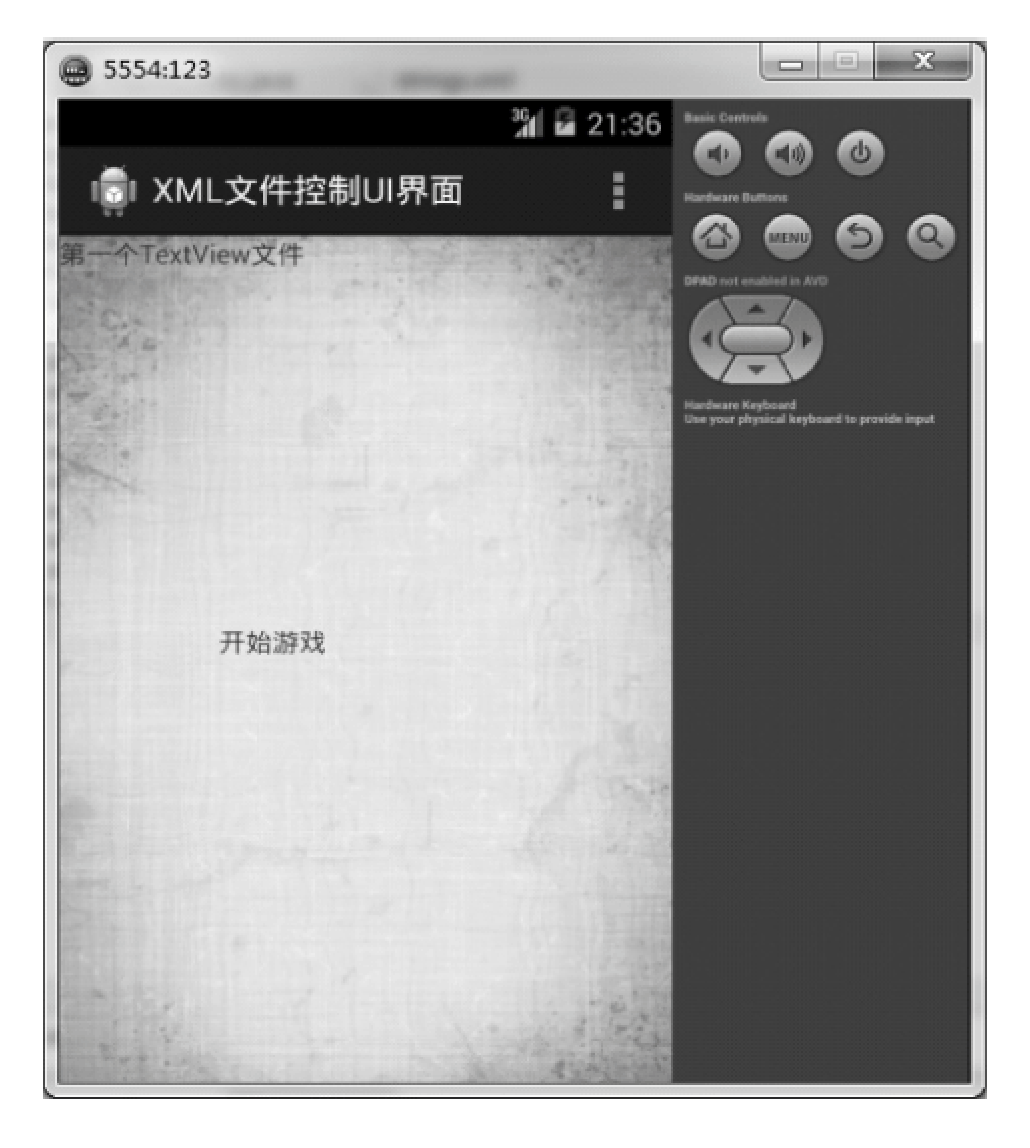


图 2-1　游戏开始界面



图 2-2　游戏界面

2.1.2　Java 代码控制 UI 界面

Android 支持像 Java Swing 那样完全通过代码控制 UI 界面。也就是所有的 UI 组件都通过 new 关键词创建出来,然后将这些 UI 控制添加到布局管理器中,从而实现用户界面。

在代码中控制 UI 界面可分为以下几个步骤。

(1) 创建布局管理器,可以是帧布局管理、表格布局管理器、线性布局管理器和相对布局管理器等,并且设置布局管理器的属性。例如,为布局管理器设置背景图片等。

(2) 创建具体的组件,可以是 TextView、ImageView、EditText 和 Button 等任何 Android 提供的组件,并且设置组件的布局和各种属性。

(3) 将创建的具体组件添加到布局管理器中。

【例 2-2】 完全通过代码实现游戏的进入界面。

其实现的步骤为:

(1) 在 Eclipse 中创建 Android 应用项目,命名为 Java_UI_test。

(2) 在新创建的项目中,打开 src/fs.java_ui_test 目录下的 MainActivity.java 文件,代码为:

```
package fs.java_ui_test;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.graphics.Color;
import android.os.Bundle;
import android.util.Log;
import android.util.TypedValue;
import android.view.Gravity;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.FrameLayout;
import android.widget.FrameLayout.LayoutParams;
import android.widget.TextView;
public class MainActivity extends Activity {
    public TextView text2;
    /** 第 1 次调用 activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        FrameLayout frameLayout = new FrameLayout(this);     //创建帧布局管理器
        frameLayout.setBackgroundDrawable(this.getResources().getDrawable(
                R.drawable.bj));                             //设置背景
        setContentView(frameLayout);                    //设置在 Activity 中显示 frameLayout
        TextView text1 = new TextView(this);
        text1.setText("在代码中控制 UI 界面");                //设置显示的文字
text1.setTextSize(TypedValue.COMPLEX_UNIT_PX,24);            //设置文字大小,单位为像素
        text1.setTextColor(Color.rgb(0,1,1));               //设置文字的颜色
        frameLayout.addView(text1);                         //将 text1 添加到布局管理器中
        text2 = new TextView(this);
        text2.setText("单击进入游戏......");                  //设置显示文字
text2.setTextSize(TypedValue.COMPLEX_UNIT_PX,24);            //设置文字大小,单位为像素
        text2.setTextColor(Color.rgb(1,0,1));               //设置文字的颜色
        LayoutParams params = new LayoutParams(
        ViewGroup.LayoutParams.WRAP_CONTENT,
    ViewGroup.LayoutParams.WRAP_CONTENT);                    //创建保存布局参数的对象
        params.gravity = Gravity.CENTER;                    //设置居中显示
        text2.setLayoutParams(params);                      //设置布局参数
        text2.setOnClickListener(new OnClickListener() {    //为 text2 添加单击事件监听
            @Override
```

```
            public void onClick(View v) {
    new AlertDialog.Builder(MainActivity.this).setTitle("系统提示")  //设置对话框的标题
    .setMessage("游戏有风险,进入需谨慎,真的要进入吗?")       //设置对话框的显示内容
                        .setPositiveButton("确定",       //为“确定”按钮添加单击事件
                            new DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface dialog,
                                        int which) {
                                Log.i("3.2","进入游戏");  //输出消息日志
                                }
                        }).setNegativeButton("退出",      //为“取消”按钮添加单击事件
                            new DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface dialog,
                                        int which) {
                                Log.i("3.2","退出游戏");  //输出消息日志
                                    finish();          //结束游戏
                                }
                            }).show();
            }
        });
        frameLayout.addView(text2);                      //将 text2 添加到布局管理器中
    }
}
```

运行程序,效果如图 2-3(a)所示。单击文字“单击进入游戏……”,将弹出如图 2-3(b)所示的提示对话框。




(a) 游戏初始界面



(b) 弹出提示对话框

图 2-3 Java 代码控制 UI 界面

说明:完全通过代码控制 UI 界面虽然比较灵活,但是其开发过程比较烦琐,而且不利于高层次的解耦,因此不推荐采用这种方式控制 UI 界面。

2.1.3 XML 和 Java 混合控制 UI 界面

完全通过 XML 布局文件控制 UI 界面，实现比较方便快捷。但是有失灵活，而完全通过 Java 代码控制 UI 界面，虽然比较灵活，但是开发过程比较烦琐。鉴于这两种方法的优缺点，下面来看另一种控制 UI 界面的方法，即使用 XML 和 Java 代码混合控制 UI 界面。

使用 XML 和 Java 代码混合控制 UI 界面。习惯上把变化小、行为比较固定的组件放在 XML 布局文件中。把变化较多、行为控制比较复杂的组件交给 Java 代码来管理。下面通过一个具体的实例来演示如何使用 XML 和 Java 代码混合控制 UI 界面。

【例 2-3】 通过 XML 和 Java 代码在窗体中纵向并列显示 3 张图片。其具体实现步骤为：

(1) 在 Eclipse 中创建 Android 应用项目，命名为 XML_Java_UI。

(2) 打开 res\layout 目录下的布局 main.xml，代码修改为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = " vertical"
    android:layout_width = "wrap_content"
    android:layout_height = "match_parent"
    android:background = "@drawable/bj1"
    android:id = "@ + id/layout" >
</LinearLayout >
```

(3) 在新创建的项目中，打开 src/fs.xml_java_ui 目录下的 MainActivity.java 文件，代码为：

```
package fs.xml_java_ui;
import android.app.Activity;
import android.os.Bundle;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.LinearLayout.LayoutParams;
public class MainActivity extends Activity {
    private  ImageView[ ] a = new ImageView[3];       //声明一个保存 ImageView 组件的数组
    private int[ ] aPath = new int[ ]{
        R.drawable.a01,R.drawable.a02,R.drawable.a03
    };                                                //声明并初始化一个保存访问图片的数组
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
                                                      //获取 XML 文件中定义的线性布局管理器
        LinearLayout layout = (LinearLayout)findViewById(R.id.layout);
        for(int i = 0;i < aPath.length;i++){
            a[i] = new ImageView(this);               //创建一个 ImageView 组件
            a[i].setImageResource(aPath[i]);          //为 ImageView 组件指定要显示的图片
            a[i].setPadding(5,5,5,5);                 //设置 ImageView 组件的内边距
                                                      //设置图片的宽度和高度
            LayoutParams params = new LayoutParams(254,168);
            a[i].setLayoutParams(params);             //为 ImageView 组件设置布局参数
```

```
                layout.addView(a[i]);                  //将 ImageView 组件添加到布局管理器中
            }
        }
    }
```

(4) 打开 AndroidManifest.xml 文件,代码修改为:

```
…
<uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="18" />
<uses-permission android:name="android.permission.SET_WALLPAPER"/>
…
```

运行程序,效果如图 2-4 所示。



图 2-4　纵向显示 3 张图片

2.2　自定义 View

在 Android 中,所有的 UI 界面都是由 View 类和 ViewGroup 类及其子类组合而成的。其中,View 类是所有 UI 组件的基类,而 ViewGroup 类是容纳这些 UI 组件的容器,其本身也是 View 类的子类。在 ViewGroup 类中,除了可以包含普通的 View 类外,还可以再次包含 ViewGroup 类。View 类和 ViewGroup 类的层次结构如图 2-5 所示。

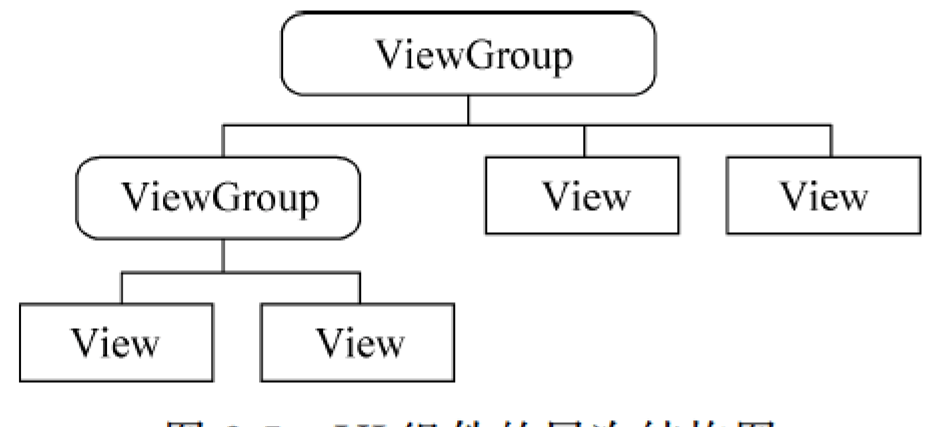


图 2-5　UI 组件的层次结构图

在一般情况下,开发 Android 应用程序的 UI 界面,不直接使用 View 类和 ViewGroup 类,而是使用这两个类的子类。例如,要显示一个图片,就可以使用 View 类的子类 ImageView。虽

然 Android 提供了很多继承了 View 类的 UI 组件，但是在实际开发时，还会出现不足以满足程序需要的情况。这时，就可以通过继承 View 类来开发自己的组件。开发自定义的 View 组件大致分为以下 3 个步骤。

（1）创建一个继承 android. view. View 类的 View 类，并且重写构造方法。

（2）根据需要重写相应的方法。通过下面的方法可以找到被重写的方法。

在代码中右击，在弹出的快捷菜单中选择“源码”|“覆盖/实现方法”命令，打开如图 2-6 所示的窗口，在该窗口的列表中显示了可以被重写的方法。只需要选中重写方法前面的复选框，并单击“确定”按钮，Eclipse 会自动重写指定的方法。通常情况下，不需要重写全部的方法。

（3）在项目的活动中，创建并实例化自定义的 View 类，并将其添加到布局管理器中。

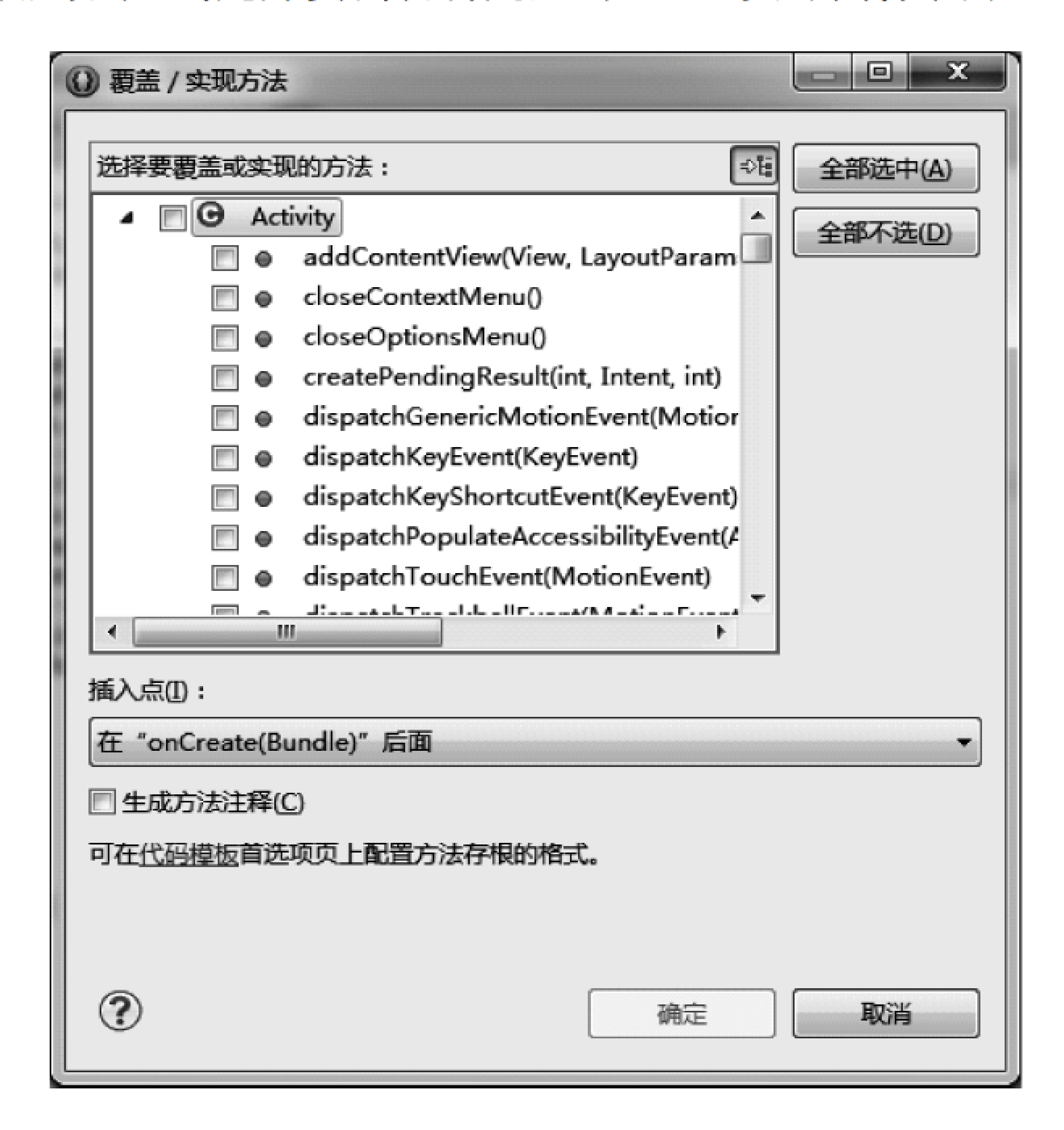


图 2-6 “覆盖/实现方法”窗口

下面通过一个实例来演示怎样开发自定义的 View 类。

【例 2-4】 通过自定义 View 组件写的一个模拟两个球不断碰撞的 View，主要由一个线程来不断更新 View 内两个球的位置，在发现两个球和墙壁发生碰撞后，改变球的逻辑参数，更新完后，调用 postInvalidate()，重绘界面。其具体实现步骤为：

（1）在 Eclipse 中创建一个 Android 应用项目，命名为 Custom_View。

（2）res\layout 目录下的 main. xml 文件采用默认代码。

（3）打开 src/fs. cutom_view 目录下的 MainActivity. java 文件，在文件中绘制两个蓝色小球及一个红色框，并实现当小球运动碰到红色框时，即可自动向上或向下运动。代码为：

```
package fs.custom_view;
import java.util.ArrayList;
import java.util.Random;
import android.app.Activity;
```

```
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Paint.Style;
import android.os.Bundle;
import android.view.View;
public class MainActivity extends Activity {
    TheScreen mScreen;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //mScreen 是自定义的 View
        mScreen = new TheScreen(this);
        setContentView(mScreen);
    }
    //为避免在程序退出后线程仍在进行,造成不必要的系统资源浪费,在 Activity 退出的时候,主
    //动将线程停止
    @Override
    public void onDestroy()
    {
        mScreen.stopDrawing();
        super.onDestroy();
    }
}
/**
 * 自定义的 View 类,为两个球的碰撞模拟
 */
class TheScreen extends View
{
    private static final String TAG = "Draw";
    //界面主线程的控制变量
    private boolean drawing = false;
    //存储当前已有的球的信息
    private ArrayList<Circle> circles;
    private Paint mPaint;
    //两个球的运动范围
    public static final int WIDTH = 300;
    public static final int HEIGHT = 400;
    public static final double PI = 3.14159265;
    Paint mPaint2 = new Paint();
    public TheScreen(Context context)
    {
        super(context);
        circles = new ArrayList<Circle>();
        //加入了两个球
        circles.add(new Circle());
        circles.add(new Circle(20,30,10));
        mPaint = new Paint();
        mPaint.setColor(Color.BLUE);
        mPaint.setAntiAlias(true);
        mPaint2.setStyle(Style.STROKE);
        mPaint2.setColor(Color.RED);
        mPaint2.setAntiAlias(true);
```

```
        //启动界面线程,开始自动更新界面
        drawing = true;
        new Thread(mRunnable).start();
    }
    private Runnable mRunnable = new Runnable() {
        //界面的主线程
        @Override
        public void run() {
            while(drawing)
            {
                try {
                    //更新球的位置信息
                    update();
                    //通知系统更新界面,相当于调用了 onDraw 函数
                    postInvalidate();
                    //界面更新的频率,这里是每 30 ms 更新一次界面
                    Thread.sleep(30);
                    //Log.e(TAG,"drawing");
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    };
    public void stopDrawing()
    {
        drawing = false;
    }
    @Override
    public void onDraw(Canvas canvas)
    {
        //在 canvas 上绘制边框
        canvas.drawRect(0,0,WIDTH,HEIGHT,mPaint2);
        //在 canvas 上绘制球
        for(Circle circle : circles)
        {
            canvas.drawCircle(circle.x,circle.y,circle.radius,mPaint);
        }
    }
    //界面的逻辑函数,主要检查球是否发生碰撞以及更新球的位置
    private void update()
    {
        if(circles.size()>1)
        {
            for(int i1 = 0; i1 < circles.size() - 1; i1++)
            {
                //当两个球发生碰撞,交换两个球的角度值
                for(int i2 = i1 + 1; i2 < circles.size(); i2++)
                    if(checkBumb(circles.get(i1),circles.get(i2)))
                    {
                        circles.get(i1).changeDerection(circles.get(i2));
                    }
            }
        }
```

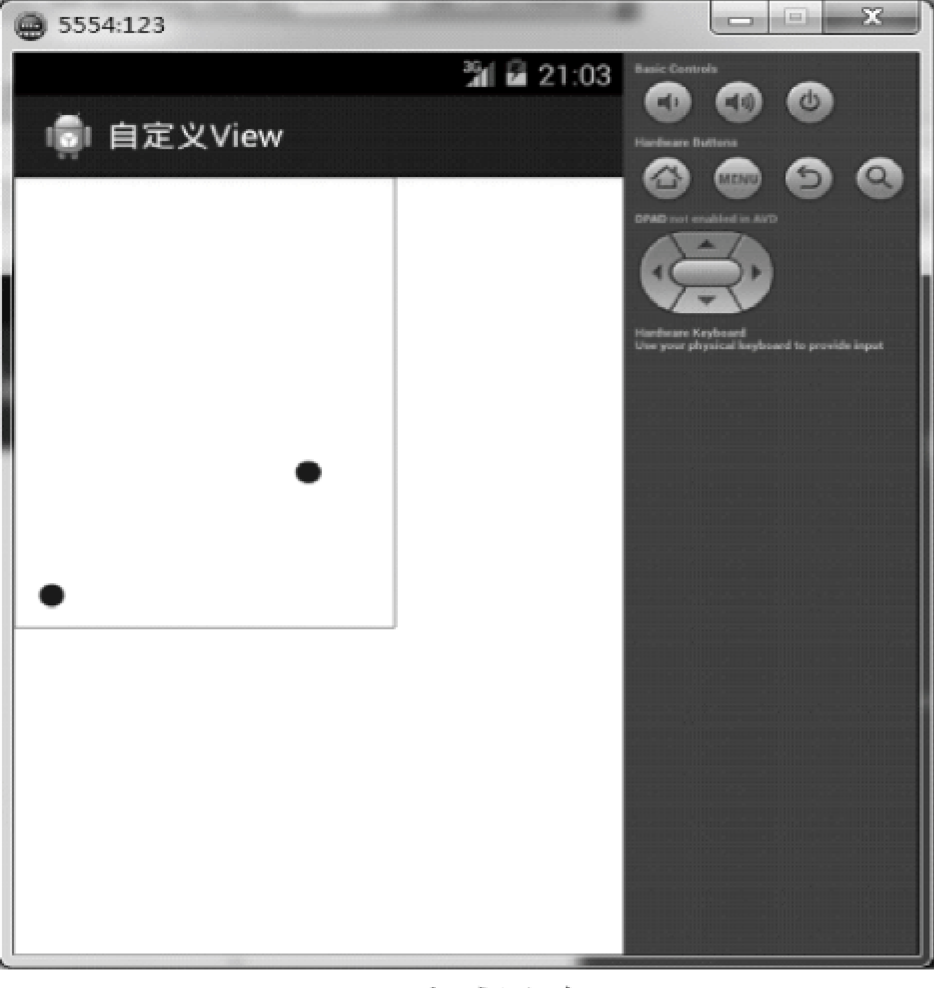
```
            //更新球的位置
            for(Circle circle: circles)
                circle.updateLocate();
        }
        private boolean checkBumb(Circle c1,Circle c2)
        {
            return (c1.x - c2.x) * (c1.x - c2.x)  +  (c1.y - c2.y) * (c1.y - c2.y) <=  (c1.radius +
    c2.radius) * (c1.radius + c2.radius);
        }
        /**
         * 自定义的 View 的内部类,存储每一个球的信息
         */
        class Circle
        {
            float x = 50;
            float y = 70;
            double angle =  (new Random().nextFloat()) * 2 * PI;
            int speed = 4;
            int radius = 10;
            public Circle() {
            }
            public Circle(float x,float y,int r)
            {
                this.x  =  x;
                this.y  =  y;
                radius  =  r;
            }
            //利用三角函数计算出球的新位置值,当与边界发生碰撞时,改变球的角度
            public void updateLocate()
            {
                x  =  x +  (float)(speed  * Math.cos(angle));
                y  =  y +  (float)(speed  * Math.sin(angle));
                if((x + radius)>= WIDTH)
                {
                    if(angle >= 0 && angle <=  (PI/2))
                        angle  =  PI  -  angle;
                    if(angle > 1.5  *  PI && angle <=  2 * PI)
                        angle  =  3  *  PI  -  angle;
                }
                if(x - radius <= 0)
                {
                    if(angle >=  PI && angle <=  1.5 * PI)
                        angle  =  3 * PI  -  angle;
                    if(angle >=  PI/2 && angle < PI)
                        angle  =  PI  -  angle;
                }
                if(y - radius <= 0 || y + radius >= HEIGHT)
                    angle  =  2 * PI  -  angle;
            }
            //两球交换角度
            public void changeDerection(Circle other)
            {
                double temp  =  this.angle;
                this.angle  =  other.angle;
```

```
            other.angle = temp;
        }
    }
}
```

运行程序，小球自动在框里运动，效果如图 2-7 所示。



(a) 小球运动1



(b) 小球运动2

图 2-7 自定义 View

2.3 Android 布局管理

在 Android 中，每个组件在窗体中都有具体的位置和大小，在窗体中摆放各种组件时，很难判断其具体位置和大小。不过，使用 Android 布局管理可以很方便地控制各组件的位置和大小。Android 提供了线性布局管理器(LinearLayout)、表格布局管理器(TableLayout)、帧布局管理器(FrameLayout)、相对布局管理器(RelativeLayout)和绝对布局管理器(AbsoluteLayout)5 种。对应于这 5 种布局管理器，Android 提供了 5 种布局方式，其中，绝对布局在 Android 2.0 中被标记为已过期，不过可以使用帧布局或相对布局替代，本节除了介绍这几个布局外，还介绍了其他布局。

2.3.1 Android 线性布局

线性布局是将其中的组件按照垂直或水平方向来布局，也就是控制放入其中的组件横向或纵向排列。在 LinearLayout 中，每一行(针对垂直排列)或每一列(针对水平排列)中只能放一个组件，并且 Android 的线性布局不会换行，当组件排列到窗体的边缘后，后面的组件将不会被显示出来。

说明：在线性布局中，排列方式由 android:orientation 属性来控制，对齐方式由 android:gravity 属性来控制。

在 Android 中，可以在 XML 布局文件中定义线性布局管理器，也可以使用 Java 代码创建，建议使用第一种。在 XML 布局文件中定义线性布局管理器时，需要使用<LinearLayout>

标记,其格式为:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:background="#aabbcc">
</LinearLayout>
```

下面对 LinearLayout 布局的常用属性进行介绍。

• android:background 属性:用于为组件设置背景,可以是背景图片,也可以是背景颜色。为组件指定背景图片时,可将准备好的图片复制到目录下,然后使用以下代码进行设置:

```
android:background="@drawable/*"
```

如果想指定背景颜色,可以使用颜色值。例如,要想指定背景颜色为黑色,可以使用以下代码:

```
android:background="#000000"
```

• android:layout_width 属性:用于设置组件的基本宽度,其可选值包括 fill_parent、match_parent 和 wrap_content。其中,fill_parent 表示该组件的宽度和父容器的宽度相同;match_parent 与 fill_parent 的作用完全相同,从 Android 2.2 开始推荐使用;wrap_content 表示该组件的宽度恰好能包裹它的内容。

注意:如果 layout_weight 值越小,则表示重要性越高,占用的空间面积也会越大,反之,亦然。

• android:layout_height 属性:用于设置组件的基本高度,其可选值包括 fill_parent、match_parent 和 wrap_content。其中,fill_parent 表示该组件的宽度和父容器的高度相同;match_parent 与 fill_parent 的作用完全相同,从 Android 2.2 开始推荐使用;wrap_content 表示该组件的高度恰好能包裹它的内容。
• android:orientation 属性:用于设置布局管理器内组件的排列方式,其可选值为 horizontal 和 vertical,其中,horizontal 表示水平排列;vertical 表示垂直排列。
• android:gravity 属性:用于设置布局管理器内组件的对齐方式,其可选值包括 top、bottom、left、right、center_vertical、fill_vertical、center_horizontal、fill_horizontal、center、fill、clip_vertical 和 clip_horizontal。这些属性值也可以同时指定,各属性之间用竖线隔开。例如,要指定组件靠下角对齐,可以使用属性值 right|bottom。

下面通过一个实例来演示线性布局管理器的使用。

【例 2-5】 采用线性布局实现绘制及显示文字的功能。其具体步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 LinearLayout_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件,删除原来默认的代码,修改代码为:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
```

```
    android:layout_height = "fill_parent"
    android:orientation = "vertical" >
<! -- 第 1 个嵌套的 LinearLayout,布局 4 个控件,注意,layout_height 为 1,数值越小,表示权重越大,后面的第 2 个嵌套布局为 2,所以第 1 个嵌套布局占用的面积大,为 2/(1 + 2) -- >
    < LinearLayout
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:layout_weight = "1"
        android:orientation = "horizontal" >
        < TextView
            android:layout_width = "wrap_content"
            android:layout_height = "fill_parent"
            android:layout_weight = "1"
            android:background = "#aa0000"
            android:gravity = "center_horizontal"
            android:text = "red" />
        < TextView
            android:layout_width = "wrap_content"
            android:layout_height = "fill_parent"
            android:layout_weight = "1"
            android:background = "#00aa00"
            android:gravity = "center_horizontal"
            android:text = "green" />
        < TextView
            android:layout_width = "wrap_content"
            android:layout_height = "fill_parent"
            android:layout_weight = "1"
            android:background = "#0000aa"
            android:gravity = "center_horizontal"
            android:text = "blue" />
        < TextView
            android:layout_width = "wrap_content"
            android:layout_height = "fill_parent"
            android:layout_weight = "1"
            android:background = "#aaaa00"
            android:gravity = "center_horizontal"
            android:text = "yellow" />
    </LinearLayout >
<! -- 第 2 个嵌套的 LinearLayout,垂直布局 4 个控件,注意,layout_height 为 2,值越小,重要性才越大,所以对比第 1 个嵌套的布局,其占用的面积为 1/(1 + 2) -- >
    < LinearLayout
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:layout_weight = "2"
        android:orientation = "vertical" >
        < TextView
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:layout_weight = "1"
            android:text = "row one"
            android:textSize = "15pt" />
```

```
            <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="row two"
                android:textSize="15pt" />
        <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="row three"
                android:textSize="15pt" />
            <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="row four"
                android:textSize="15pt" />
        </LinearLayout>
    </LinearLayout>
```

运行程序，效果如图 2-8 所示。

在本实例中，如果将 android:orientation 属性的值设置为 horizontal，则采用水平线性布局，效果如图 2-9 所示。



图 2-8　垂直线性布局



图 2-9　水平线性布局

2.3.2　Android 表格布局

表格布局与常见的表格类似，它以行、列的形式来管理放入其中的 UI 组件。表格布局使用＜TableLayout＞标记定义，在表格布局中，可以添加多个＜TableRow＞标记，每个＜TableRow＞标记占用一行，由于＜TableRow＞标记也是容器，所以在该标记中还可添加

其他组件，在<TableRow>标记中，每添加一个组件，表格就会增加一列。在表格布局中，列可以被隐藏，也可以被设置为伸展的，从而填充时可利用的屏幕空间，也可以设置强制收缩，直到表格匹配屏幕大小。

说明：如果在表格布局中直接向<TableLayout>中添加 UI 组件，那么这个组件将独占一行。

在 Android 中，可以在 XML 布局文件中定义表格布局管理器，也可以使用 Java 代码创建。推荐使用在 XML 布局文件中定义表格布局管理器。在 XML 布局文件中定义表格布局管理器的格式为：

```
<TableLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent" >
</TableLayout>
```

TableLayout 继承了 LinearLayout，因此它完全支持 LinearLayout 所支持的全部 XML 属性。此外 TableLayout 支持如表 2-1 所示的 XML 属性。

表 2-1　TableLayout 支持的 XML 属性

XML 属性	描　述
android:collapseColumns	设置需要被隐藏的列序号(序号从 0 开始)，多个列序号之间用逗号“,”分隔
android:shrinkColumns	设置允许被收缩的列序号(序号从 0 开始)，多个列序号之间用逗号“,”分隔
android:stretchColumns	设置允许被拉伸的列序号(序号从 0 开始)，多个列序号之间用逗号“,”分隔

下面通过一个实例来演示表格布局的用法。

【例 2-6】 应用表格布局实现几个按钮控件的排列。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 TableLayout_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件，删除默认代码，采用表格布局实现按钮的排列。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<TableLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
    <TableRow >
        <Button
            android:id = "@ + id/button1"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:layout_weight = "1"
            android:text = "Button1" />
    </TableRow>
    <TableRow >
        <Button
            android:id = "@ + id/button2"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:layout_weight = "1"
```

```
            android:text = "Button2" />
    </TableRow>
    <TableRow>
        <Button
            android:id = "@ + id/button3"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_weight = "1"
            android:text = "Button3" />
    </TableRow>
    <TableRow>
        <Button
            android:id = "@ + id/button4"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_weight = "1"
            android:text = "Button4" />
    </TableRow>
    <View
        android:layout_height = "2dip"
        android:background = "#FF909090" />
    <TableRow>
        <Button
            android:id = "@ + id/button5"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_weight = "1"
            android:text = "Button5" />
        <Button
            android:id = "@ + id/button6"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_weight = "1"
            android:text = "Button6" />
        <Button
            android:id = "@ + id/button7"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_weight = "1"
            android:text = "Button7" />
        <Button
            android:id = "@ + id/button8"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_weight = "1"
            android:text = "Button8" />
    </TableRow>
</TableLayout>
```

运行程序，效果如图 2-10 所示。

注意：如果设置 android:collapse=“1”，则该 TableLayout 里的 TableRow 的列为 1，即



图 2-10　表格布局 1

第二列(从 0 开始计算)。

下面通过改变其对应属性,即按钮排列的顺序不相同,代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<TableLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:stretchColumns = "0,1,2">
    <TableRow>
      <Button
        android:id = "@ + id/button1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "Button1"
        android:layout_column = "0"/>
      <Button
        android:id = "@ + id/button2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "Button2"
        android:layout_column = "1" />
      </TableRow>
      <TableRow>
        <Button
          android:id = "@ + id/button3"
          android:layout_width = "wrap_content"
          android:layout_height = "wrap_content"
```

```
                android:text = "Button3"
                android:layout_column = "1"/>
              < Button
                android:id = "@ + id/button4"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:text = "Button4"
                android:layout_column = "1"/>
              </TableRow >
              < TableRow >
                < Button
                android:id = "@ + id/button5"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:text = "Button5"
                android:layout_column = "2"/>
                </TableRow >
</TableLayout >
```

运行程序,效果如图 2-11 所示。



图 2-11　表格布局 2

2.3.3　Android 帧布局

在帧布局管理器中,每加入一个组件,都将创建一个空白的区域,通常称为一帧,这些帧都会根据 gravity 属性执行自动对齐。默认情况下,帧布局是从屏幕的左上角(0,0)坐标点开始布局,多个组件层叠排序,后面的组件覆盖前面的组件。

在 Android 中，可以在 XML 布局文件中定义帧布局管理器，也可以使用 Java 代码来创建。推荐使用在 XML 布局文件中定义帧布局管理器。在 XML 布局文件中，定义帧布局管理器可以使用<FrameLayout>标记，格式为：

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
</FrameLayout>
```

FrameLayout 支持的常用 XML 属性如表 2-2 所列。

表 2-2　FrameLayout 支持的常用 XML 属性

XML 属性	描　　述
android:foreground	设置该帧布局容器的前景图像
android:foregroundGravity	定义绘制前景图像的 gravity 属性，也就是前景图像显示的位置

【例 2-7】 应用帧布局居中显示层叠的长方形。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 FrameLayout_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件，删除布局文件的默认代码，修改后的代码为：

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
      android:id="@+id/frameLayout1"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      xmlns:android="http://schemas.android.com/apk/res/android"
      android:background="#FFF"
      android:foreground="@drawable/ic_launcher"
      android:foregroundGravity="bottom|right">
      <!-- 添加居中显示的蓝色背景的 TextView,将显示在最下层 -->
    <TextView
        android:text="蓝色背景的 TextView"
        android:id="@+id/textView1"
        android:background="#FF0000FF"
        android:layout_gravity="center"
        android:layout_width="440px"
        android:layout_height="300px"/>
      <!-- 添加居中显示的天蓝色背景的 TextView,将显示在中间层 -->
    <TextView
        android:text="天蓝色背景的 TextView"
        android:id="@+id/textView2"
        android:layout_width="360px"
        android:layout_height="200px"
        android:background="#FF0077FF"
        android:layout_gravity="center" />
      <!-- 添加居中显示的水蓝色背景的 TextView,将显示在最上层 -->
    <TextView
        android:text="水蓝色背景的 TextView"
```

```
            android:id = "@ + id/textView3"
            android:layout_width = "240px"
            android:layout_height = "120px"
            android:background = "#FF00B4FF"
            android:layout_gravity = "center"/>
</FrameLayout>
```

运行程序,效果如图 2-12 所示。



图 2-12　帧布局

2.3.4　Android 相对布局

相对布局(RelativeLayout)是指一个 ViewGroup 以相对位置显示它的子视图(View)元素,一个视图可以指定相对于它的兄弟视图的位置(例如在给定视图的左边或下面)或相对于 RelativeLayout 的特定区域的位置。例如底部对齐,或中间偏左。

相对布局是设计用户界面的有力工具,因为它消除了嵌套视图组。如果用户发现使用了多个嵌套的 LinearLayout 视图组,可以考虑使用一个 RelativeLayout 视图组。

在 Android 中,可以在 XML 布局文件中定义相对布局管理器,也可以使用 Java 代码创建。建议使用 XML 布局文件中定义相对布局管理器。在 XML 布局文件中,定义相对布局管理器可以使用<RelativeLayout>标记,其格式为:

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
```

```
android:paddingLeft = "@dimen/activity_horizontal_margin"
android:paddingRight = "@dimen/activity_horizontal_margin"
android:paddingTop = "@dimen/activity_vertical_margin"
tools:context = ".MainActivity" >
```

在进行相对布局时用到的属性很多，首先来看属性值只为 true 或 false 的属性，如表 2-3 所示。

表 2-3　相对布局中只取 true 或 false 的属性

属 性 名 称	描　　述
android:layout_centerHorizontal	当前控件位于父控件的横向中间位置
android:layout_centerVertical	当前控件位于父控件的纵向中间位置
android:layout_centerInParent	当前控件位于父控件的中央位置
android:layout_alignParentBottom	当前控件底端与父控件底端对齐
android:layout_alignParentLeft	当前控件左侧与父控件左侧对齐
android:layout_alignParentRight	当前控件右侧与父控件右侧对齐
android:layout_alignParentTop	当前控件顶端与父控件顶端对齐
android:layout_alignWithParentIfMissing	参照控件不存在或不可见时参照父控件

接下来再来看属性值为其他控件 id 的属性，如表 2-4 所示。

表 2-4　相对布局中取值为其他控件 id 的属性及说明

属 性 名 称	描　　述
android:layout_toRightOf	使当前控件位于给定 ID 控件右边
android:layout_toLeftOf	使当前控件位于给定 ID 控件左边
android:layout_above	使当前控件位于给定 ID 控件之上
android:layout_below	使当前控件位于给定 ID 控件之下
android:layout_alignTop	使当前控件的上边界与给定 ID 对齐
android:layout_alignBottom	使当前控件的下边界与给定 ID 对齐
android:layout_alignLeft	使当前控件的左边界与给定 ID 对齐
android:layout_alignRight	使当前控件的右边界与给定 ID 对齐

最后要介绍的是属性值以像素为单位的属性及说明，如表 2-5 所示。

表 2-5　相对布局中取值为像素的属性及说明

属 性 名 称	描　　述
android:layout_marginLeft	当前控件左侧的留白
android:layout_marginRight	当前控件右侧的留白
android:layout_marginTop	当前控件上方的留白
android:layout_marginBottom	当前控件下方的留白

下面通过一个实例来演示相对布局管理器的用法。

【例 2-8】 利用相对布局实现一个登录界面。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 RelativeLayout_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件，删除默认代码，代码修改为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc" >
    <EditText
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignParentTop = "true"
        android:layout_alignParentRight = "true"
        android:layout_toRightOf = "@ + id/tv_username"
        android:id = "@ + id/txt_username"/>
    <EditText
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_below = "@ + id/txt_username"
        android:layout_alignLeft = "@ + id/txt_username"
        android:layout_alignParentRight = "true"
        android:id = "@ + id/txt_password"/>
    <TextView
        android:id = "@ + id/tv_username"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "用户名称"
        android:layout_alignParentTop = "true"
        android:layout_alignParentLeft = "true"
        android:layout_marginTop = "14dp"/>
    <TextView
        android:id = "@ + id/tv_password"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignBottom = "@ + id/txt_password"
        android:layout_alignLeft = "@ + id/btn_cacel"
        android:text = "用户密码" />
    <Button
        android:id = "@ + id/btn_login"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignRight = "@ + id/txt_password"
        android:layout_below = "@ + id/txt_password"
        android:layout_marginTop = "33dp"
        android:text = "登录" />
    <Button
        android:id = "@ + id/btn_cacel"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
```

```
        android:layout_alignBaseline = "@ + id/btn_login"
        android:layout_alignBottom = "@ + id/btn_login"
        android:layout_alignLeft = "@ + id/tv_username"
        android:text = "取消" />
</RelativeLayout>
```

运行程序,效果如图 2-13 所示。



图 2-13　绝对布局

2.4　Android 基本布局综合实例

下面将通过综合实例的实现过程,来讲解使用基本布局控件的方法。

【例 2-9】　实现 Android 的四种基本布局。其实现操作为:

(1) 在 Eclipse 新建一个 Android 应用项目,命名为 Layout_test。

(2) 编写布局文件,打开 res/Layout 目录下的 main.xml 文件,其代码修改为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
    < Button android:id = "@ + id/button0"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "使用 FrameLayout" />
    < Button android:id = "@ + id/button1"
```

```
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:text = "使用 RelativeLayout" />
        < Button android:id = "@ + id/button2"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:text = "使用 LinearLayout 和 RelativeLayout" />
        < Button android:id = "@ + id/button3"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:text = "使用 TableLayout" />
</LinearLayout >
```

以上代码为一个典型的 LinearLayout 布局样式。

(3) 编写主文件,打开 src/fs. layout_test 包下的 MainActivity. java 文件,该文件主要用于调用公用文件来实现具体的功能。其实现代码为:

```
public class MainActivity extends Activity
{
    OnClickListener listener0  =  null;
    OnClickListener listener1  =  null;
    OnClickListener listener2  =  null;
    OnClickListener listener3  =  null;
    Button button0;
    Button button1;
    Button button2;
    Button button3;
    /** 第 1 次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        listener0  =  new OnClickListener()
        {
            public void onClick(View v)
            {
            Intent intent0  =  new Intent(MainActivity.this,ActivityFrameLayout.class);
                setTitle("FrameLayout");
                startActivity(intent0);
            }
        };
        listener1  =  new OnClickListener()
        {
            public void onClick(View v)
            {
            Intent intent1  =  new Intent(MainActivity.this,ActivityRelativeLayout.class);
                startActivity(intent1);
            }
        };
        listener2  =  new OnClickListener()
        {
```

```
            public void onClick(View v)
            {
                setTitle("这是在 ActivityLayout");
                Intent intent2 = new Intent(MainActivity.this,ActivityLayout.class);
                startActivity(intent2);
            }
        };
        listener3 = new OnClickListener()
        {
            public void onClick(View v)
            {
                setTitle("TableLayout");
                Intent intent3 = new Intent(MainActivity.this,ActivityTableLayout.class);
                startActivity(intent3);
            }
        };
        setContentView(R.layout.main);
        button0 = (Button) findViewById(R.id.button0);
        button0.setOnClickListener(listener0);
        button1 = (Button) findViewById(R.id.button1);
        button1.setOnClickListener(listener1);
        button2 = (Button) findViewById(R.id.button2);
        button2.setOnClickListener(listener2);
        button3 = (Button) findViewById(R.id.button3);
        button3.setOnClickListener(listener3);
    }
}
```

以上代码中，函数 setContentView(R.layout.main)用于实现 Activity 和 main.xml 的关联；button0、button1、button2、button3 代表了 4 个按钮，在上述代码中这 4 个按钮实现了引用，并给按钮设置了单击监听器，每一个监听器都跳转到一个新的 Activity。

(4) 在 res/Layout 目录下新建一个名为 activityframelayout.xml 的文件，该文件实现第 1 个按钮 button0。单击按钮 button0 即会显示一个风景图，此界面为一个 FrameLayout 布局。定义了这幅风景图的显示样式，即在 FrameLayout 布局中添加了一个图片显示组件 ImageView 元素，实现代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< FrameLayout android:id = "@ + id/left"
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
    < ImageView android:id = "@ + id/photo" android:src = "@drawable/bg"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content" />
</FrameLayout >
```

(5) 在 res/Layout 目录下新建一个名为 relativelayout.xml 的文件。该文件用于单击第 2 个按钮 button1 后的处理动作。单击按钮 button1 后会显示要求输入用户名的表单，此功能是通过 RelativeLayout 实现的。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- Demonstrates using a relative layout to create a form -->
<RelativeLayout
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent" android:layout_height = "wrap_content"
    android:padding = "10dip">
    <TextView android:id = "@ + id/label" android:layout_width = "fill_parent"
        android:layout_height = "wrap_content" android:text = "请输入用户名: " />
    <! -- 这个 EditText 放置在 id 为 label 的 TextView 的下边 -->
    <EditText android:id = "@ + id/entry" android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:background = "@android:drawable/editbox_background"
        android:layout_below = "@id/label" />
    <! -- 取消按钮和容器的右边齐平,并且设置左边的边距为 10dip -->
    <Button android:id = "@ + id/cancel" android:layout_width = "wrap_content"
        android:layout_height = "wrap_content" android:layout_below = "@id/entry"
        android:layout_alignParentRight = "true"
        android:layout_marginLeft = "10dip" android:text = "取消" />
    <! -- 确定按钮在取消按钮的左侧,并且和取消按钮的高度齐平 -->
    <Button android:id = "@ + id/ok" android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_toLeftOf = "@id/cancel"
        android:layout_alignTop = "@id/cancel" android:text = "确定" />
</RelativeLayout>
```

(6) 实现第 3 个按钮 button2 处理动作。在实例中,单击 button2 按钮即显示一系列文本,此功能是通过 LinearLayout 和 RelativeLayout 联合实现的。具体实现如下:

① 在 res/Layout 目录下新建一个名为 left.xml 的 RelativeLayout 布局文件,实现第 a 组第 a 项和第 a 组和第 b 项,实现代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<RelativeLayout
android:id = "@ + id/left"
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
    <TextView android:id = "@ + id/view1"
        android:layout_width = "fill_parent"
        android:layout_height = "50px" android:text = "第 a 组第 a 项" />
    <TextView android:id = "@ + id/view2"
        android:layout_width = "fill_parent"
        android:layout_height = "50px" android:layout_below = "@id/view1"
        android:text = "第 a 组第 b 项" />
</RelativeLayout>
```

② 在 res/Layout 目录下新建一个名为 right.xml 的 RelativeLayout 布局文件,实现第 b 组第 a 项和第 b 组和第 b 项,实现代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<RelativeLayout android:id = "@ + id/right"
```

```
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
    < TextView android:id = "@ + id/right_view1"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content" android:text = "第 b 组第 a 项" />
    < TextView android:id = "@ + id/right_view2"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:layout_below = "@id/right_view1" android:text = "第 b 组第 b 项" />
</RelativeLayout >
```

③ 实现 Layout 与 Activity 的关联。即实现一个 Layout 和一个 Activity 的关联，而此 Layout 是在 XML 文件中被定义的。在 Activity 中，为使用方便可自行构建一个 Layout。根据需要在 res/fs.layout_test 包下新建一个名为 ActivityLayout.java，代码为：

```
public class ActivityLayout extends Activity
{
    /** 第 1 次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        LinearLayout layoutMain = new LinearLayout(this);
        layoutMain.setOrientation(LinearLayout.HORIZONTAL);
        setContentView(layoutMain);
         LayoutInflater inflate = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_
SERVICE);
        RelativeLayout layoutLeft = (RelativeLayout) inflate.inflate(
                R.layout.left,null);
        RelativeLayout layoutRight = (RelativeLayout) inflate.inflate(
                R.layout.right,null);
        RelativeLayout.LayoutParams relParam = new RelativeLayout.LayoutParams(
                RelativeLayout.LayoutParams.WRAP_CONTENT,
                RelativeLayout.LayoutParams.WRAP_CONTENT);
        layoutMain.addView(layoutLeft,100,100);
        layoutMain.addView(layoutRight,relParam);
    }
}
```

(7) 设计单击第 4 个按钮 button3 的处理动作。单击按钮 button3 即会显示一个整齐排列的表单，此功能是通过 TableLayout 实现的。在 res/Layout 目录下新建一个名为 activitytablelayout.xml 的文件，代码为：

```
< TableLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent" android:layout_height = "fill_parent"
    android:stretchColumns = "1">
    < TableRow >
        < TextView android:text = "用户名:" android:textStyle = "bold"
            android:gravity = "right" android:padding = "3dip" />
```

```
        <EditText android:id="@+id/username" android:padding="3dip"
            android:scrollHorizontally="true" />
    </TableRow>
    <TableRow>
        <TextView android:text="密码:" android:textStyle="bold"
            android:gravity="right" android:padding="3dip" />
        <EditText android:id="@+id/password" android:password="true"
            android:padding="3dip" android:scrollHorizontally="true" />
    </TableRow>
    <TableRow android:gravity="right">
        <Button android:id="@+id/cancel"
            android:text="取消" />
        <Button android:id="@+id/login"
            android:text="登录" />
    </TableRow>
</TableLayout>
```

（8）实现对应的布局，根据需要分别在 res/fs. layout_test 包下新建名为 ActivityRelativieLayout. java、ActivityTableLayout. java 文件、ActivityFrameLayout. java 文件，代码分别为：

ActivityRelativieLayout. java 文件代码为：

```
public class ActivityRelativeLayout extends Activity
{
    /** 第 1 次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.relativelayout);
    }
}
```

ActivityTableLayout. java 文件代码为：

```
public class ActivityTableLayout extends Activity
{
    /** 第 1 次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activitytablelayout);
    }
}
```

ActivityFrameLayout. java 文件代码为：

```
public class ActivityFrameLayout extends Activity
{
    /** 第 1 次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
```

```
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activityframelayout);
    }
}
```

运行程序,效果如图 2-14 所示。



图 2-14　基本布局实例效果

2.5　Android 其他布局

在 Android 中除了介绍的四种布局外,还有其他布局,下面分别介绍。

2.5.1　Android 网格布局

网格布局是 Android 4.0 版本才有的,在低版本使用该布局需要导入对应支撑库。GridLayout 将整个容器划分成 rows×columns 个网格,每个网格可以放置一个组件。还可以设置一个组件横跨多少列、多少行。不存在一个网格放多个组件的情况。

GridLayout 的布局策略简单分为以下 3 个部分:

首先,它与 LinearLayout 布局一样,也分为水平和垂直两种方式,默认是水平布局的,一个控件挨着一个控件从左到右依次排列,但是通过指定 android:columnCount 设置列数的属性后,控件会自动换行进行排列。另一方面,对于 GridLayout 布局中的子控件,默认按照 wrap_content 的方式设置其显示,这只需要在 GridLayout 布局中显式声明即可。

其次,若要指定某控件显示在固定的行或列,只需设置该子控件的 android:layout_row 和 android:layout_column 属性即可,但是需要注意:android:layout_row="0"表示从第 1 行开始,android:layout_column="0"表示从第 1 列开始,这与编程语言中一维数组的赋值

情况类似。

最后，如果需要设置某控件跨越多行或多列，只需要将该子控件的 android：layout_rowSpan 或者 layout_columnSpan 属性设置为数值，再设置 layout_gravity 属性为 fill 即可，前一个设置表明该控件跨越的行数或列数，后一个设置表明该控件填满所跨越的整行或整列。

下面通过一个案例来演示 GridLayout 类的使用。

【例 2-10】 利用 GridLayout 布局编写的简易计算器。其具体实现步骤为：

（1）在 Eclipse 中创建一个 Android 应用项目，命名为 GridLayout_test。

（2）打开 res\layout 目录下的 main. xml 文件，用于在屏幕中布局计算器格局。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< GridLayout
        xmlns:android = "http://schemas.android.com/apk/res/android"
        android:layout_width = "match_parent"
        android:layout_height = "match_parent"
        android:orientation = "horizontal"
        android:useDefaultMargins = "true"
        android:alignmentMode = "alignBounds"
        android:columnOrderPreserved = "false"
        android:rowCount = "5"
        android:columnCount = "4"
        android:background = "#000000">
    < Button
        android:id = "@ + id/one"
        android:text = "1"/>
    < Button
        android:id = "@ + id/two"
        android:text = "2"/>
      < Button
        android:id = "@ + id/three"
        android:text = "3"/>
    < Button
        android:id = "@ + id/devide"
        android:text = "/"/>
    < Button
        android:id = "@ + id/four"
        android:text = "4"/>
    < Button
        android:id = "@ + id/five"
        android:text = "5"/>
    < Button
        android:id = "@ + id/six"
        android:text = "6"/>
    < Button
        android:id = "@ + id/multiply"
        android:text = " × "/>
    < Button
        android:id = "@ + id/seven"
        android:text = "7"/>
    < Button
        android:id = "@ + id/eight"
```

```
        android:text = "8"/>
    < Button
        android:id = "@ + id/nine"
        android:text = "9"/>
    < Button
        android:id = "@ + id/minus"
        android:text = " - "/>
    < Button
        android:id = "@ + id/zero"
        android:layout_columnSpan = "2"
        android:layout_gravity = "fill"
        android:text = "0"/>
    < Button
        android:id = "@ + id/point"
        android:text = "."/>
    < Button
        android:id = "@ + id/plus"
        android:layout_rowSpan = "2"
        android:layout_gravity = "fill"
        android:text = " + "/>
    < Button
        android:id = "@ + id/equal"
        android:layout_columnSpan = "3"
        android:layout_gravity = "fill"
        android:text = " = "/>
</GridLayout >
```

运行程序,效果如图 2-15 所示。



图 2-15　计算器界面

2.5.2　Android 切换卡

切换卡(TabWidget)是一种相对复杂的布局管理器,通过多个标签来切换显示不同的内容,一个 TabWidget 主要是由一个 TabHost 来存放多个 Tab 标签容器,再在 Tab 容器中加入其他控件,通过 addTab 方法可以增加新的 Tab,这些除了在 XML 文件中布置好控件外,当然还需要在 Java 文件中处理好事件的逻辑。

TabWidget 继承自 LinearLayout,是线性布局的一种,除了继承自父类的属性和方法,

在 FrameLayout 类中包含了自己特有的属性和方法，如表 2-6 所示。

表 2-6　TabWidget 常用的属性

属　性	描　述
android:divider	可绘制对象，被绘制在选项卡窗口间充当分割物
android:tabStripEnabled	确定是否在选项卡绘制
android:tabStripLeft	被用来绘制选项卡下面的分割线左边部分的可视化对象
android:tabStripRight	被用来绘制选项卡下面的分割线右边部分的可视化对象

下面通过一个实例来演示 TabWidget 控件的用法。

【例 2-11】 利用 TabWidget 实现几个标签页面切换。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 TabWidget_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件，代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- 定义了 TabHost 布局,其 id 必须为 @android:id/tabhost -->
< TabHost xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@android:id/tabhost"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
    <! -- 定义了 3 个切换卡的整体布局方式 -->
    < LinearLayout
        android:orientation = "vertical"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent">
          <! -- 定义了切换卡 TabWidget,其 id 必须为@android:id/tabs -->
        < TabWidget
            android:id = "@android:id/tabs"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content" />
        <! -- 定义了切换卡内 FrameLayout 布局,其 id 必须为@android:id/tabcontent -->
        < FrameLayout
            android:id = "@android:id/tabcontent"
            android:layout_width = "fill_parent"
            android:layout_height = "fill_parent">
            < TextView
                android:id = "@ + id/textview1"
                android:layout_width = "fill_parent"
                android:layout_height = "fill_parent"
                android:text = "这是一个 tab" />
                < TextView
                    android:id = "@ + id/textview2"
                    android:layout_width = "fill_parent"
                    android:layout_height = "fill_parent"
                    android:text = "这是另一个 tab" />
                  < TextView
                    android:id = "@ + id/textview3"
                    android:layout_width = "fill_parent"
                    android:layout_height = "fill_parent"
```

```
                    android:text="这是第 3 个 tab" />
            </FrameLayout>
        </LinearLayout>
</TabHost>
```

(3) 打开 src\fs. TabWidget_test 包下的 MainActivity 文件，在文件中实现 3 个标签的切换，并弹出对应的对话框。其代码为：

```
public class MainActivity extends TabActivity
{
    //声明 TabHost 对象
    TabHost mTabHost;
    /** 第 1 次调用 activity */
        @Override
        public void onCreate(Bundle savedInstanceState)
        {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
         //取得 TabHost 对象
         mTabHost = getTabHost();
         /* 为 TabHost 添加标签 */
        //新建一个 newTabSpec(newTabSpec)
         //设置其标签和图标(setIndicator)
         //设置内容(setContent)
         mTabHost.addTab(mTabHost.newTabSpec("tab_test1")
             .setIndicator("TAB 1",getResources().getDrawable(R.drawable.b))
             .setContent(R.id.textview1));
         mTabHost.addTab(mTabHost.newTabSpec("tab_test2")
             .setIndicator("TAB 2",getResources().getDrawable(R.drawable.b1))
             .setContent(R.id.textview2));
         mTabHost.addTab(mTabHost.newTabSpec("tab_test3")
             .setIndicator("TAB 3",getResources().getDrawable(R.drawable.b2))
             .setContent(R.id.textview3));
         //设置 TabHost 的背景颜色
         mTabHost.setBackgroundColor(Color.argb(150,22,70,150));
         //设置 TabHost 的背景图片资源
         mTabHost.setBackgroundResource(R.drawable.bj1);
         //设置当前显示哪一个标签
         mTabHost.setCurrentTab(0);
         //标签切换事件处理,setOnTabChangedListener
         mTabHost.setOnTabChangedListener(new OnTabChangeListener()
        {
             @Override
            public void onTabChanged(String tabId)
            {
                 Dialog dialog = new AlertDialog.Builder(MainActivity.this)
                     .setTitle("提示")
                     .setMessage("当前选中: "+tabId+"标签")
                     .setPositiveButton("确定",new DialogInterface.OnClickListener()
                     {
                     public void onClick(DialogInterface dialog, int whichButton)
```

```
                                {
                                        dialog.cancel();
                                }
                            }).create();                        //创建按钮
                            dialog.show();
                    }
            });
    }  }
```

运行程序，效果如图 2-16 所示。



图 2-16　切换卡界面

第3章 Android 控件设计

控件是 Android 程序设计的基本组成单位，通过使用控件可以高效地开发 Android 应用程序。所以，熟练掌握控件的使用是合理、有效地进行 Android 程序开发的重要前提。

3.1 Widget 控件实例

在 Eclipse 中创建一个新的工程，命名为 Widget_test，用于对各种常见控件进行学习。其具体实现步骤为：

（1）新建项目。单击"文件"|"新建"|Android Application Project，打开 New Android Application 对话框，如图 3-1 所示。

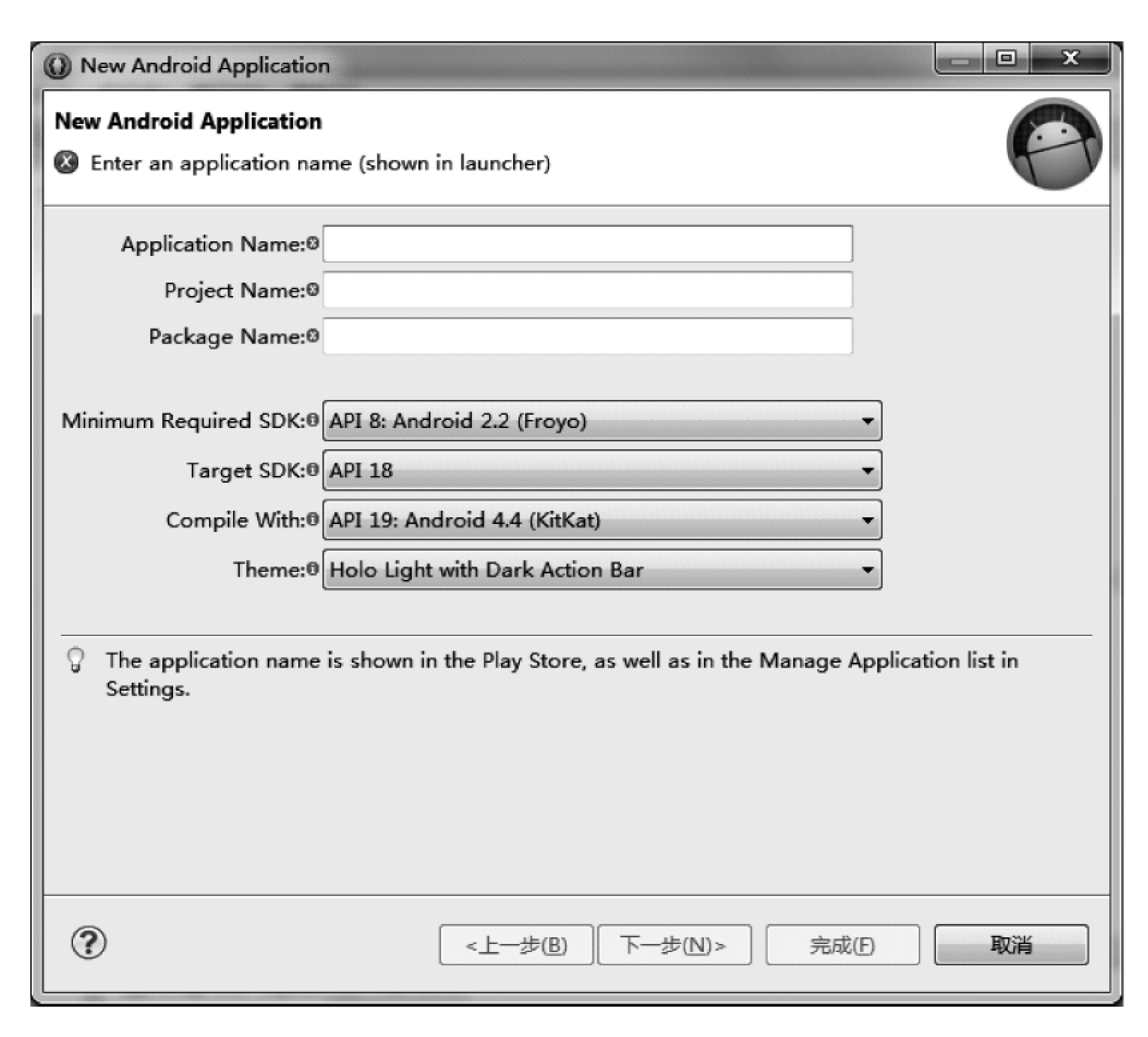


图 3-1　新建项目

（2）输入工程名称 Widget_test，在 Location 后的文本框中输入工程的保存路径，并按默认值新建应用项目。

MainActivity. java 文件是当前应用程序的入口类 MainActivity 的定义文件。双击 MainActivity. java，其生成代码为：

```
package fs.widget_test;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        //该菜单用于增加项目的操作栏
        getMenuInflater().inflate(R.menu.main,menu);
        return true;
    }
}
```

其中 onCreate()方法中的 setContentView(R. layout. main)表明 MainActivity 使用的用户界面 UI 文件为 main. xml。

双击 main. xml 文件，发现 Eclipse 为其提供了 Graphical Layout 和 main. xml 两种浏览方式。其中 Graphical Layout 方式为以图形方式浏览 main. xml 文件，其效果等同于 main. xml 在手机设备上运行的效果；main. xml 方式为以代码方式浏览 main. xml 文件。这两种方式是等效的，都可以对 main. xml 文件进行编辑和查看。双击 main. xml 文件，其自动生成代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity" >
    <TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "@string/hello_world" />
</RelativeLayout>
```

该文件表明，当前 main. xml 文件所使用的布局为 RelativeLayout 布局，该布局自动填满整个手机屏幕。在该布局中，放置了一个 TextView 控件，该 TextView 显示的内容为 @string/hello，表示 string. xml 文件中定义的 hello 变量的内容。打开 values 目录下的 string. xml 文件，其自动生成代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < string name = "app_name"> Widget_test </string >
    < string name = "action_settings"> Settings </string >
    < string name = "hello_world"> Hello world! </string >
</resources >
```

单击 main. xml 的 Graphical Layout 浏览方式，可查看当前文件的图形化效果，如图 3-2 所示。



图 3-2　文件的图形化效果

程序开发人员可以在该图形方式下，将左侧的各种组件直接拖放到屏幕上形成自己想要的布局，也可以直接修改 main. xml 文件的代码。

3.2　Android 文件类控件

在 Android 中，文本控件主要包括 TextView 控件和 EditText 控件。

3.2.1　Android 文本框

在 Android 中，文本框使用 TextView 表示，用于在屏幕上显示文本。这与 Java 中的文本框组件不同，它相当于 Java 中的标签，也就是 JLable。值得说明的是，Android 中的文本框组件可以显示单行文本、多行文本以及带图像的文本。

在 Android 中，可以使用两种方法向屏幕中添加文本框，一种是通过在 XML 布局文件中使用<TextView>标记添加，另一种是在 Java 文件中，通过 new 关键字创建。建议使用第一种方法，也就是通过<TextView>标记在 XML 布局文件中添加。

TextView 支持的常用 XML 属性如表 3-1 所示。

表 3-1 TextView 的 XML 属性及说明

XML 属性	相关方法	描述
android:autoLink	setAutoLinkMask(int)	是否将符合指定格式的文本转换为可单击的超链接形式
android:cursorVisible	setCursorVisible(Boolean)	设置该文本框的光标是否可见
android:drawableBottom	setCompoundDrawablesWithIntrinsicBounds(Drawable,Drawable,Drawable,Drawable)	在文本框内文本的底端绘制指定图像
android:drawableLeft	setCompoundDrawablesWithIntrinsicBounds(Drawable,Drawable,Drawable,Drawable)	在文本框内文本的左端绘制指定图像
android:drawablePadding	setCompoundDrawablesWithIntrinsicBounds(Drawable,Drawable,Drawable,Drawable)	在文本框内文本的间隙绘制指定图像
android:drawableRight	setCompoundDrawablesWithIntrinsicBounds(Drawable,Drawable,Drawable,Drawable)	在文本框内文本的右端绘制指定图像
android:drawableTop	setCompoundDrawablesWithIntrinsicBounds(Drawable,Drawable,Drawable,Drawable)	在文本框内文本的顶端绘制指定图像
android:editable		设置该文本是否允许编辑
android:ellipsize		设置当显示的文本超过了TextView的长度时如何处理文本内容
android:gravity	setGravity(int)	设置文本框内文本的对齐方式
android:height	setHeight(int)	设置该文本框的高度(单位为pixel)
android:hint	setHint(int)	设置当该文本框内容为空时,文本框内默认显示的提示文本
android:minHeight	setMinHeight(int)	指定该文本框的最小高度,单位为 pixel
android:maxHeight	setMaxHeight(int)	指定该文本框的最大高度,单位为 pixel
android:minWidth	setMinWidth(int)	指定该文本框的最小宽度,单位为 pixel
android:maxWidth	setMaxWidth(int)	指定该文本框的最大宽度,单位为 pixel
android:lines	setlines(int)	设置该文本框默认占几行
android:MinLines	setMinLines(int)	设置该文本框最少占几行
android:MaxLines	setMaxLines(int)	设置该文本框最多占几行
android:password	setTransformationMethod(TransformationMethod)	设置文本框为一个密码框,以点代替字符
android:phoneNumber	setKeyListener(KeyListener)	设置该文本框只能接收电话号码
android:scrollHorizontally	setHorizontallyScrolling(Boolean)	设置文本框不够显示全部内容时是否允许水平滚动

续表

XML 属性	相关方法	描述
android:selectAllOnFocus	setSelectAllOnFocus(Boolean)	如果文本框的内容可选,设置当它获得焦点时是否自动选中所有文本
android:singleLine	setTransformationMethod	设置文本框是否为单行模式,如设为 true,则不会换行
android:shadowColor	setShadowLayer(float,float,float,int)	设置文本框内文本的阴影颜色
android:shadowDx	setShadowLayer(float,float,float,int)	设置文本框内文本的阴影在水平方向的偏移
android:shadowDy	setShadowLayer(float,float,float,int)	设置文本框内文本的阴影在垂直方向的偏移
android:shadowRadius	setShadowLayer(float,float,float,int)	设置文本框内文本的阴影角度
android:text	setText(CharSequence)	设置文本框内文本的内容
android:textColor	setTextColor(ColorStateList)	设置文本框内文本的颜色
android:textColorHighlight	setHighlightColor(int)	设置文本框内文本被选中时的颜色
android:textScaleX	setTextScaleX(float)	设置文本框内文本在水平方向上的缩放因子
android:textSize	setTextSize(float)	设置文本框内文本的字体大小
android:textStyle	setTypeface(Typeface)	设置文本框内文本的字体风格
android:typeface	setTypeface(Typeface)	设置文本框内文本的字体
android:width	setWidth(int)	设置文本框宽度,单位为 pixel

表 3-1 中 android:autoLink 属性值是几个属性值的一个或几个,多个属性值之间用竖线隔开。

下面通过一个实例来演示 TextView 的用法。为文本框中的 E-mail 地址添加超链接、显示带图像的文本、显示不同颜色的单行文本和多行文本。其具体操作步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 TextView_test。

(2) 打开 res\layout 目录下的 main.xml 文件,其代码修改为:

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:background = "@drawable/bj"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity" >
    <TextView
        android:id = "@ + id/textView4"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:autoLink = "email"
        android:height = "50px"
        android:text = "@string/hello_world" />
    <!-- 添加一个新的 TextView 控件,用于显示带图像的文本 -->
```

```
    < TextView
        android:id = "@ + id/textView1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/textView2"
        android:layout_centerVertical = "true"
        android:drawableTop = "@drawable/ic_launcher"
        android:text = "带图片的 TextView" />
    <! -- 设置为可以显示多行文本(默认) -->
    < TextView
        android:id = "@ + id/textView2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/textView4"
        android:layout_below = "@ + id/textView4"
        android:layout_marginTop = "32dp"
        android:text = "多行文本: 蓝天上蓝得一丝云彩都看不见,深蓝和浅蓝混合的天显得格外
宁静"
        android:textColor = " # 09F"
        android:textSize = "20px"
        android:width = "300px" />
    <! -- 一个设置为只能显示单行文本 -->
    < TextView
        android:id = "@ + id/textView3"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignParentLeft = "true"
        android:layout_below = "@ + id/textView4"
        android:singleLine = "true"
        android:text = "单行文本: 蓝天上蓝得一丝云彩都看不见,深蓝和浅蓝混合的天显得格外
宁静"
        android:textColor = " # f00"
        android:textSize = "20px"
        android:width = "300px" />
</RelativeLayout >
```

运行程序,效果如图 3-3 所示。



图 3-3　应用 TextView 显示多种多样的文本

3.2.2 Android 编辑框

在 Android 中，编辑框使用 EditText 表示，用于在屏幕上显示文本输入框，这与 Java 中的文本框组件功能类似。值得说明的是，Android 中的编辑框组件可以输入单行文本，也可以输入多行文本，而且还可以输入指定格式的文本(如密码、电话号码、E-mail 等)。

在 Android 中，可以使用两种方法向屏幕中添加编辑框，一种是通过在 XML 布局文件中使用<EditText>标记添加，另一种是在 Java 文件中，通过 new 关键字创建。建议使用第一种方法，也就是通过<EditText>标记在 XML 布局文件中添加。

由于 EditText 类是 TextView 的子类，所以对于表 3-1 中列出的 XML 属性，同样适用于 EditText 组件。值得注意的是，在 EditText 组件中，android:inputType 属性可以帮助输入法显示合适的类型。例如，要添加一个密码框，可以将 android:inputType 属性设置为 textPassword。

表 3-2 列出了 EditText 类继承自 TextView 类中的常用属性及对应的方法说明。

表 3-2 EditText 的 XML 属性及说明

XML 属性	相关方法	说明
android:cursorVisible	setcursorVisible (boolean)	设置光标是否可见，默认为可见
android:lines	setLines(int)	通过设置固定的行数来决定 EditText 的高度
android:maxLines	setMaxLines(int)	设置最大行数
android:minLines	setMinLines(int)	设置最小行数
android:password	setTransformationMethod (TransformationMethod)	设置文本框中的内容是否显示为密码
android:phoneNumber	setKeyListener(KeyListener)	设置文本框中的内容只能是电话号码
android:scrollHorizontally	setHorizontallyScrolling(boolean)	设置文本框是否可以水平地进行滚动
android:selectAllOnFocus	setSelectAllOnFocus(boolean)	如果文本内容可选中，则当文本框获得焦点时自动选中全部文本内容
android:shadowColor	setShadowLayer (float, float, float, int)	为文本框设置指定颜色的阴影
android:shadowDx	setShadowLayer (float, float, float, int)	为文本框设置阴影的水平偏移，为浮点数
android:shadowDy	setShadowLayer (float, float, float, int)	为文本框设置阴影的垂直偏移，为浮点数
android:shadowRadius	setShadowLayer (float, float, float, int)	为文本框设置阴影的半径，为浮点数
android:singleLine	setTransformationMethod (TransformationMethod)	设置文本框为单行模式
android:maxLength	setFilters(InputFilter)	设置最大显示长度

下面通过一个实例说明 EditText 的用法。本实例主要用于实现一个简单的登录界面，在界面中，如果输入正确的用户名和密码，单击“确定”按钮，将出现一个提示“恭喜您登录成功!”，否则将提示“请重新输入用户名或密码!”。单击“清空”按钮，则会清空所填写的姓名

和密码内容。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 EditText_test。

(2) 打开 res\layout 下的 main. xml 文件，代码修改为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
    <LinearLayout
        android:orientation = "vertical"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:background = "#aabbcc"
        android:paddingLeft = "5dip"
        android:paddingRight = "5dip"
        android:paddingTop = "5dip">
        <LinearLayout
            android:id = "@ + id/LinearLayout01"
            android:orientation = "horizontal"
            android:layout_width = "fill_parent"
            android:layout_height = "fill_parent">
            <TextView
                android:text = "用户名："
                android:id = "@ + id/TextView02"
                android:textColor = "#222444"
                android:layout_width = "wrap_content"
                android:layout_height = "40dip"
                android:layout_marginLeft = "5dip"
                android:textSize = "18dip"
                android:gravity = "center_vertical"/>
            <EditText
                android:id = "@ + id/EditTextuid"
                android:singleLine = "true"
                android:layout_width = "fill_parent"
                android:layout_height = "wrap_content"
                android:layout_marginLeft = "0dip"
                android:text = "e11"/>
        </LinearLayout>
        <LinearLayout
            android:id = "@ + id/LinearLayout02"
            android:orientation = "horizontal"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content">
            <TextView
                android:text = "密      码："
                android:id = "@ + id/TextView03"
                android:textColor = "#222222"
                android:layout_width = "wrap_content"
                android:layout_height = "40dip"
                android:layout_marginLeft = "5dip"
                android:textSize = "18dip"
```

```
                android:gravity = "center_vertical"/>
            < EditText
                android:id = "@ + id/EditTextPwd"
                android:singleLine = "true"
                android:layout_width = "fill_parent"
                android:layout_height = "wrap_content"
                android:text = "112233">
            </EditText >
        </LinearLayout >
        < LinearLayout
            android:id = "@ + id/LinearLayout03"
            android:orientation = "horizontal"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content">
            < Button
                android:text = " 登录 "
                android:id = "@ + id/loginLog"
                android:layout_width = "75dip"
                android:layout_height = "40dip"
                android:textSize = "18dip"
                android:gravity = "center"/>
            < Button
                android:text = " 清空 "
                android:id = "@ + id/loginClear"
                android:layout_width = "75dip"
                android:layout_height = "40dip"
                android:textSize = "18dip"
                android:gravity = "center"/>
        </LinearLayout >
    </LinearLayout >
</LinearLayout >
```

（3）打开 src/fs. edittext_test 包下 MainActivity. java 文件，在文件中实现按钮的响应，代码为：

```
package fs.edittext_test;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends Activity {
    /** 第 1 次调用 activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button bLogin  = (Button)this.findViewById(R.id.loginLog);            //"登录"按钮
        Button bClear = (Button)this.findViewById(R.id.loginClear);           //"清空"按钮
        final EditText eUid = (EditText)this.findViewById(R.id.EditTextuid);  //用户名
        final EditText eMima = (EditText)this.findViewById(R.id.EditTextPwd); //密码
        bLogin.setOnClickListener                                 //添加"登录"按钮监听器
```

```
            (
                new OnClickListener()
                {
                    public void onClick(View v)
                    {
                        String strUid = eUid.getText().toString().trim();
                        String strPwd = eMima.getText().toString().trim();
                        if(strUid.equals("e11")&&strPwd.equals("112233"))
                        {
                            Toast.makeText(MainActivity.this,"恭喜您登录成功!",Toast.LENGTH_
SHORT).show();
                        }
                        else
                        {
                            Toast.makeText(MainActivity.this,"请输入重新输入用户名或密码!",
Toast.LENGTH_SHORT).show();
                        }
                    }
                }
            );
            bClear.setOnClickListener                        //添加"清空"按钮监听器
            (
                    new OnClickListener()
                    {
                        public void onClick(View v) {
                            eUid.setText("");                //设置用户名为空
                            eMima.setText("");               //设置密码为空
                        }
                    }
            );
        }
    }
```

运行程序，默认效果如图 3-4(a)所示，当单击界面中的“清空”按钮时，效果如图 3-4(b)所示，当输入不正确的用户名或密码时，单击界面中的“确定”按钮时，效果如图 3-4(c)所示，当输入正确的用户名和密码时，单击界面中的“确定”按钮时，效果如图 3-4(d)所示。

3.2.3 Android 自动提示文本框

除了基本的文本控件外，在 Android 中还提供了一个自动提示文本框 AutoCompleteTextView，所谓的自动提示就是在文本框中输入文字时，会显示可能的关键字让用户来选择。在其他系统下完成此功能可能非常麻烦，但是在 Android 中是非常容易达到的。

AutoCompleteTextView 类继承自 EditText 类，位于 android.widget 包下。自动提示文本框的外观与图片文本框没有任何区别，只是当用户输入某些文字时，会自动出现下拉菜单，显示与用户输入文字相关的信息，用户直接单击需要的文字，便可自动填写到文本控件中。

对于自动提示文本框的设置可以在 XML 文件中使用属性进行设置，也可以在 Java 中通过方法进行设置，表 3-3 列出了 AutoCompleteTextView 类支持的常用属性及方法。



(a) 默认效果

(b) 清空界面效果



(c) 输入错误的用户名或密码后效果



(d) 输入正确的用户名和密码后效果

图 3-4　EditText 控件的用法

表 3-3　AutoCompleteTextView 支持的常用属性及方法

XML 属性	方　　法	描　　述
android:completionHint	setCompletionHint(CharSequence)	设置出现在下拉菜单中的提示标题
android:completionThreshold	setThreshold(int)	设置用户至少输入几个字符才会显示提示
android:dropDownHeight	setDropDownHeight(int)	设置下拉菜单的高度
android:dropDownHorizontalOffset		设置下拉菜单与文本框之间的水平偏移。下拉菜单默认与文本框左对齐
android:dropDownVerticalOffset		设置下拉菜单与文本框之间的垂直偏移。下拉菜单默认紧跟文本框
android:dropDownWidth	setDropWidth(int)	设置下拉菜单的宽度
android:popupBackground	setDropDownBackgroundResource(int)	设置下拉菜单的背景

使用 AutoCompleteTextView 很简单，只要为它设置一个 Adapter，该 Adapter 封装了 AutoCompleteTextView 预设的提示文本。

下面通过一个实例来演示 AutoCompleteTextView 类的用法。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 AutoCompleteTextView_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件，代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
    < LinearLayout
        android:layout_width = "0px"
        android:layout_height = "0px"
        android:focusable = "true"
        android:focusableInTouchMode = "true"/>
    < AutoCompleteTextView
        android:hint = "请输入文字进行搜索"
        android:layout_height = "wrap_content"
        android:layout_width = "match_parent"
        android:id = "@ + id/autoCompleteTextView1"/>
    < Button
        android:text = "搜索"
        android:id = "@ + id/button1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"/>
</LinearLayout >
```

(3) 打开 src\fs.autoncompletetextview_test 包下的 MainActivity.java 文件，在该文件中实现了 autocompletetextview 从 sharepreference 中读取历史记录并显示的功能，当没有任何输入时，提示最新的 5 项历史记录(这里可以加个条件，当有历史记录时才显示)。代码为：

```
package fs.autocompletetextview_test;
import android.app.Activity;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.View.OnFocusChangeListener;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
public class MainActivity extends Activity implements
        OnClickListener {
    private AutoCompleteTextView autoTv;
```

```
/** 第 1 次调用 activity 活动 */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    autoTv = (AutoCompleteTextView) findViewById(R.id.autoCompleteTextView1);
    initAutoComplete("history",autoTv);
    Button search = (Button) findViewById(R.id.button1);
    search.setOnClickListener(this);
}
@Override
public void onClick(View v) {
    //这里可以设定：当搜索成功时,才执行保存操作
    saveHistory("history",autoTv);
}
/**
  * 初始化 AutoCompleteTextView,最多显示 5 项提示,使
  * AutoCompleteTextView 在一开始获得焦点时自动提示
  * @param field 保存在 sharedPreference 中的字段名
  * @param auto 要操作的 AutoCompleteTextView
  */
private void initAutoComplete(String field,AutoCompleteTextView auto) {
    SharedPreferences sp = getSharedPreferences("network_url",0);
    String longhistory = sp.getString("history","nothing");
    String[] hisArrays = longhistory.split(",");
    ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_dropdown_item_1line,hisArrays);
    //只保留最近的 50 条的记录
    if(hisArrays.length > 50){
        String[] newArrays = new String[50];
        System.arraycopy(hisArrays,0,newArrays,0,50);
        adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_dropdown_item_1line,newArrays);
    }
    auto.setAdapter(adapter);
    auto.setDropDownHeight(350);
    auto.setThreshold(1);
    auto.setCompletionHint("最近的 5 条记录");
    auto.setOnFocusChangeListener(new OnFocusChangeListener() {
        @Override
        public void onFocusChange(View v,boolean hasFocus) {
            AutoCompleteTextView view = (AutoCompleteTextView) v;
            if (hasFocus) {
                    view.showDropDown();
            }
        }
    });
}
/**
  * 把指定 AutoCompleteTextView 中内容保存到 sharedPreference 中指定的字符段
  * @param field 保存在 sharedPreference 中的字段名
```

```
         * @param auto 要操作的 AutoCompleteTextView
         */
        private void saveHistory(String field,AutoCompleteTextView auto) {
            String text = auto.getText().toString();
            SharedPreferences sp = getSharedPreferences("network_url",0);
            String longhistory = sp.getString(field,"nothing");
            if (!longhistory.contains(text + ",")) {
                StringBuilder sb = new StringBuilder(longhistory);
                sb.insert(0,text + ",");
                sp.edit().putString("history",sb.toString()).commit();
            }
        }
    }
```

运行程序,效果如图 3-5 所示。



图 3-5　自动提示文本框

3.2.4　Android 多选项自动提示文本框

在控件 AutoCompleteTextView 中一次只能选择一个选项并且支持模糊查询功能,而在 MultiAutoCompleteTextView 控件中是可以选择多个选项并且也支持模糊查询的功能。其主要方法主要有:

- enoughToFilter():当文本长度超过阈值时过滤。
- performValidation():代替验证整个文本,这个子类方法验证每个单独的文字标记。
- setTokenizer(MultiAutoCompleteTextView.Tokenizer t):用户正在输入时,tokenizer 设置将用于确定文本相关的范围内。

下面通过一个实例来演示 MultiAutoCompleteTextView 控件的用法。其具体实现步骤为：

（1）在 Eclipse 中创建一个 Android 应用项目，命名为 MultiAutoCompleteTextView_test。

（2）打开 res\layout 目录下的 main.xml 文件，代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
    <MultiAutoCompleteTextView
        android:text = ""
        android:id = "@ + id/multiAutoCompleteTextView1"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"/>
</LinearLayout>
```

（3）打开 src\fs.multiautocompletetextview_test 包下的 MainActivity.java 文件，在该文件中实现多个选项选择的查询功能。代码为：

```
package fs.multiautocompletetextview_test;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.MultiAutoCompleteTextView;
import android.widget.TextView;
import android.widget.AdapterView.OnItemClickListener;
public class MainActivity extends Activity {
    private static final String[] COUNTRIES = new String[] { "Android study","Android world",
            "Android test","Android view","Android Multi" };
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_dropdown_item_1line,COUNTRIES);
         MultiAutoCompleteTextView textView = (MultiAutoCompleteTextView) findViewById
(R.id.multiAutoCompleteTextView1);
        textView.setAdapter(adapter);
        textView.setThreshold(1);
        textView.setTokenizer(new MultiAutoCompleteTextView.CommaTokenizer());
        textView.setOnItemClickListener(new OnItemClickListener() {
            public void onItemClick(AdapterView<?> arg0,View arg1,int arg2,
                    long arg3) {
                Log.v("textView.setOnItemClickListener",""
                        + ((TextView) arg1).getText());
```

```
            }
        });
    }
}
```


运行程序,效果如图 3-6 所示。

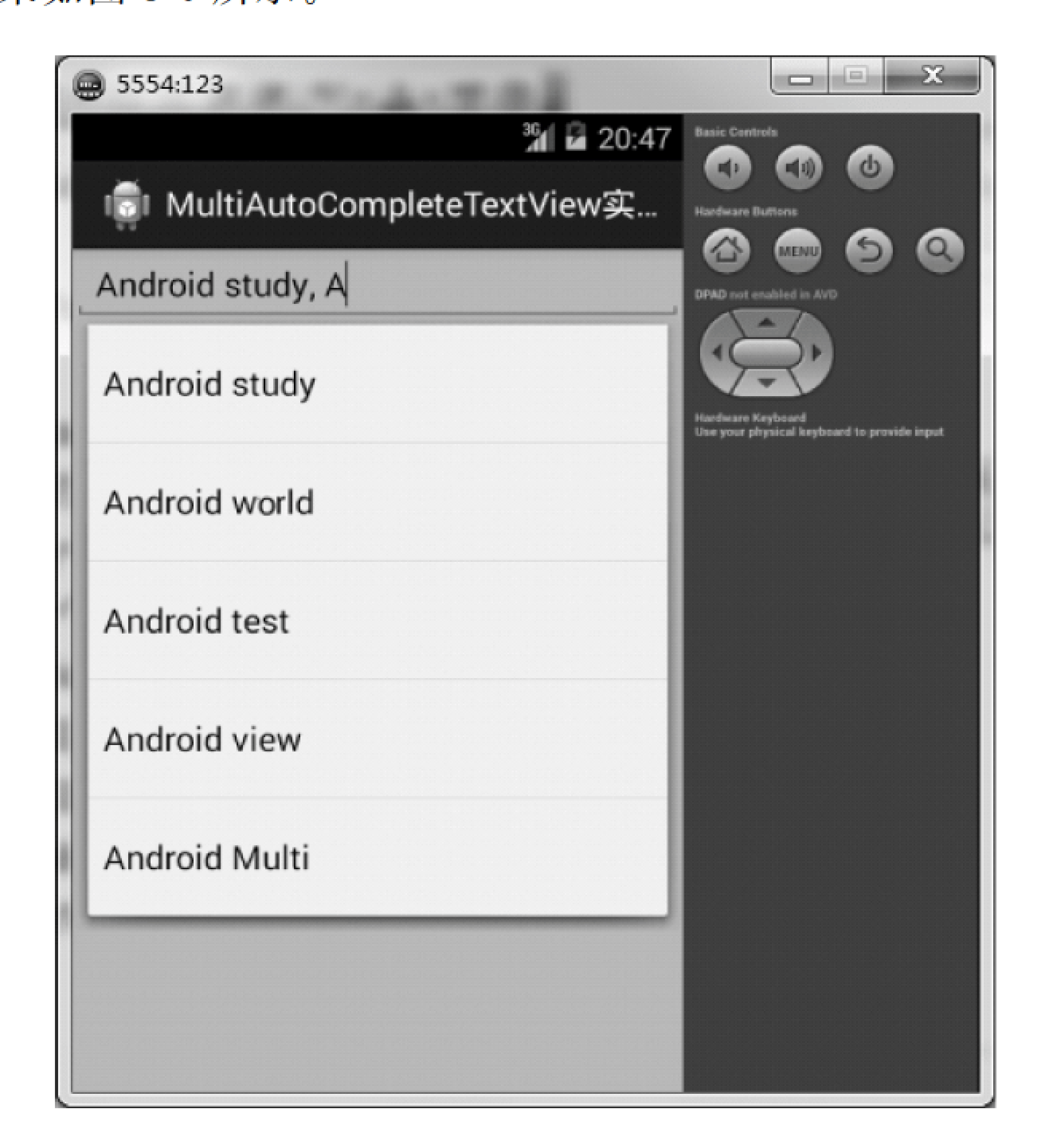


图 3-6　多选项自动文本提示框效果

3.3　Android 按钮类控件

在 Android 中,提供了一些按钮类的控件,主要包括普通按钮、图片按钮、单选按钮和复选按钮,下面分别给予介绍。

3.3.1　Android 普通按钮

Button 控件继承自 TextView 类,用户可以对 Button 控件进行按下或单击等操作。在 Android 中,可以使用两种方法向屏幕中添加按钮,一种是通过在 XML 布局文件中使用<Button>标记添加,另一种是在 Java 文件中,通过 new 关键字创建。建议采用第一种方法,也就是通过<Button>标记在 XML 布局文件中添加。在 XML 布局文件中添加普通按钮的基本格式为:

```
<Button
    android:id = "@ + id/button1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_alignTop = "@ + id/textView1"
```

```
    android:layout_marginLeft = "46dp"
    android:layout_toRightOf = "@ + id/textView1"
    android:text = "Button" />
```

在屏幕上添加按钮后，还需要为按钮添加单击事件监听器，才能让按钮发挥其特有的用途。在 Android 中，提供了两种为按钮添加单击事件监听器的方法，一种是在 Java 代码中完成的。例如，在 Activity 的 onCreate()方法中完成，代码为：

```
import android.view.View.OnClickListener;
import android.widget.Button;

Button login = (Button)findViewByld(R.id.login);         //通过 ID 获取布局文件中添加的按钮
login.setOnClickListener(new onClickListener(){          //为按钮添加单击事件监听器
    @Override
    public void onClick(View v){
        //编写要执行的动作代码
        }
});
```

另一种是在 Activity 中编写包含 View 类型参数的方法，并且将要触发的动作代码放在该方法中，然后在布局文件中，通过 android:onClick 属性指定对应的方法名实现。例如，在 Activity 中编写一个 butClick()方法，关键代码为：

```
public void butClick(View view)
{
    //编写要执行的动作代码
}
```

那么即可在布局文件中通过 android:onClick="butClick"为按钮添加单击事件监听器。

下面通过一个实例来演示 Button 控件的用法。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Button_test。

(2) 打开 res\layout 目录下的 main.xml 文件，代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
    < TextView
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "@string/hello_world" />
    < Button
        android:text = "Button"
        android:id = "@ + id/button1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"/>
    < Button
        android:text = "Button"
```

```
            android:id = "@ + id/button2"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:drawableTop = "@drawable/ic_launcher"
            android:drawablePadding = "5px"/>
        <Button
            android:text = "Button"
            android:id = "@ + id/button3"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:drawableBottom = "@drawable/ic_launcher"
            android:drawablePadding = "5px"/>
        <Button
            android:text = "Button"
            android:id = "@ + id/button4"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:drawableLeft = "@drawable/ic_launcher"
            android:drawablePadding = "5px"/>
        <Button
            android:text = "Button"
            android:id = "@ + id/button5"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:drawableRight = "@drawable/ic_launcher"
            android:drawablePadding = "5px"/>
    </LinearLayout>
```

属性 android:drawableTop 的功能是用于定义图像资源放在文字的哪个方向，而属性 android:drawablePadding 是定义图像与文字的间距为多少。

(3) 打开 src\fs. button_test 包下的 MainActivity. java 文件，用于实现按钮的监听功能。代码为：

```
package fs.button_test;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
public class MainActivity extends Activity {
    private static final String TAG = "MainActivity";
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button button1 = (Button) this.findViewById(R.id.button1);
        button1.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
                Log.v(TAG,"单击了按钮!");
```

```
            }
        });
    }
}
```

运行程序,效果如图 3-7 所示。



图 3-7 普通按钮的实例

3.3.2 Android 图片按钮

ImageButton 控件继承自 ImageView 类,ImageButton 控件与 Button 控件的主要区别在于 ImageButton 中没有 text 属性,即按钮将显示的是图片而不是文本。在 ImageButton 控件中设置按钮显示的图片可通过 android:src 属性来设置,也可通过 setImageResource(int)方法来设置。

默认情况下,ImageButton 同 Button 一样具有背景色,当按钮处于不同的状态(如按下等)时,背景色也会随之变化。当 ImageButton 所显示的图片不能完全覆盖背景色时,这种显示效果将会非常糟糕,所以使用 ImageButton 一般要将背景色设置为其他图片或直接设置为透明的。

下面通过一个实例来演示 ImageButton 控件的用法。其具体操作步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 ImageButton_test。

(2) 打开 res\layout 目录下的 main.xml 文件,在该文件中声明一个 TextView 控件,用于显示提示单击第几个按钮;声明两个 ImageButton 控件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
```

```
        android:orientation = "vertical"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:background = "#aabbcc">
    <TextView
            android:id = "@+id/TextView1"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"/>
        <ImageButton
            android:id = "@+id/ImageButton1"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:background = "@drawable/d1"/>
        <ImageButton
            android:id = "@+id/ImageButton2"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:background = "@drawable/d2"/>
    </LinearLayout>
```

(3) 打开 src\fs.imagebutton_test 包下的 MainActivity.java 文件，在文件中实现当单击 ImageButton 控件时实现图像的更换。代码为：

```
package fs.imagebutton_test;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ImageButton;
import android.widget.TextView;
public class MainActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final TextView tv = (TextView)this.findViewById(R.id.TextView1);
        final ImageButton but1 = (ImageButton)this.findViewById(R.id.ImageButton1);
        final ImageButton but2 = (ImageButton)this.findViewById(R.id.ImageButton2);
        but1.setOnClickListener
        (
                new OnClickListener()
                {
                    public void onClick(View v)
                    {
                        tv.setText("单击的是第 1 个 ImageButton");
but1.setImageDrawable(getResources().getDrawable(R.drawable.d3));
but2.setImageDrawable(getResources().getDrawable(R.drawable.d4));
                    }
                }
```

```
        );
        but2.setOnClickListener
        (
                new OnClickListener()
                {
                    public void onClick(View v)
                    {
                        tv.setText("单击的是第 2 个 ImageButton");
but2.setImageDrawable(getResources().getDrawable(R.drawable.d4));
but1.setImageDrawable(getResources().getDrawable(R.drawable.d1));
                    }
                }
        );
    }
}
```

运行程序，效果如图 3-8 所示。





图 3-8　ImageButton 按钮效果

3.3.3　Android 单选按钮

在默认情况下，单选按钮(RadioButton)显示为一个圆形图标，并且在该图标旁边放置一些说明性文字，而在程序中，一般将多个单选按钮放置在按钮组(RadioGroup)中，使这些单选按钮表现出某种功能，当用户选中某个单选按钮后，按钮组中的其他按钮将被自动取消选中状态。RadioButton 类是 Button 的子类，所以单选按钮可以直接使用 Button 支持的各种属性。

在 Android 中，可使用两种方法向屏幕中添加单选按钮，一种是通过在 XML 布局文件中使用<RadioButton>标记添加，另一种是在 Java 文件中通过 new 关键字创建。建议使用第一种方法，即通过<RadioButton>在 XML 布局文件中添加。在 XML 布局文件中添加单选按钮的格式为：

```
<RadioButton
    android:id = "@ + id/ID 号"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_alignParentLeft = "true"
    android:layout_below = "@ + id/textView1"
    android:layout_marginTop = "37dp"
    android:checked = "true|false"
    android:text = "显示文本" />
```

RadioButton 组件的 android:checked 属性用于指定选中状态，属性值为 true 时，表示选中；属性值为 false 时，表示不选中，默认为 false。

在通常情况下，RadioButton 控件需要与 RadioButton 控件一起使用，组成一个单选按钮组。在 XML 文件中，添加 RadioGroup 组件的格式为：

```
<RadioGroup
    android:id = "@ + id/radioGroup1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_alignRight = "@ + id/radioButton1"
    android:layout_below = "@ + id/radioButton1"
    android:layout_marginTop = "45dp" >
```

下面通过一个实例来演示 RadioButton 控件的用法。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 RadioButton_test。

(2) 打开 res\layout 目录下的 main.xml 文件，在布局文件中声明一个 RadioGroup 控件，在该控件中定义 4 个 RadioButton 控件，还声明一个 TextView 控件及一个 Button 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
<RadioGroup
    android:id = "@ + id/radioGroup1"
    android:layout_height = "wrap_content"
    android:layout_width = "match_parent"
    android:orientation = "horizontal">
        <RadioButton
            android:id = "@ + id/radio1"
            android:tag = "bj"
            android:layout_height = "wrap_content"
            android:text = "黑龙江"
            android:layout_width = "wrap_content"/>
        <RadioButton
            android:id = "@ + id/radio2"
            android:tag = "sh"
```

```
            android:layout_height = "wrap_content"
            android:text = "哈尔滨"
            android:layout_width = "wrap_content"/>
        <RadioButton
            android:id = "@ + id/radio3"
            android:tag = "sz"
            android:layout_height = "wrap_content"
            android:text = "香港"
            android:layout_width = "wrap_content"
            android:checked = "true"/>
        <RadioButton
            android:id = "@ + id/radio4"
            android:tag = "gz"
            android:layout_height = "wrap_content"
            android:text = "深圳"
            android:layout_width = "wrap_content"/>
    </RadioGroup>
    <TextView
        android:text = "TextView"
        android:id = "@ + id/textView1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"/>
    <Button
        android:text = "Button"
        android:id = "@ + id/button2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"/>
</LinearLayout>
```

(3) 打开 src\fs. RadioButton_test 包下的 MainActivity. java 文件，在文件中实现单选按钮的选择功能，并将选择结果显示在 TextView 中。代码为：

```
package fs.radiobutton_test;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.RadioGroup.OnCheckedChangeListener;
public class MainActivity extends Activity {
    private Button button2;
    private TextView textView1;
    private RadioGroup radioGroup1;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
```

```
            radioGroup1 = (RadioGroup) this.findViewById(R.id.radioGroup1);
            button2 = (Button) this.findViewById(R.id.button2);
            textView1 = (TextView) this.findViewById(R.id.textView1);
            radioGroup1.setOnCheckedChangeListener(new OnCheckedChangeListener() {
                public void onCheckedChanged(RadioGroup arg0, int arg1) {
                    textView1.setText(((RadioButton) MainActivity.this.findViewById(arg1))
                            .getText().toString());
                }
            });
            button2.setOnClickListener(new OnClickListener() {
                public void onClick(View arg0) {
                    Log.v("!!",((RadioButton) MainActivity.this.findViewById(radioGroup1
                            .getCheckedRadioButtonId())).getTag().toString());
                }
            });
        }
    }
```

运行程序,效果如图 3-9 所示。



图 3-9　单选按钮效果

3.3.4　Android 复选按钮

在默认情况下,复选按钮(CheckBox)显示一个方块图标,并且在该图标旁边放置一些说明性文字。与单选按钮唯一不同的是复选按钮可以进行多选设置,每一个复选按钮都提供“选中”和“不选中”两种状态。CheckBox 类又是 Button 的子类,所以复选按钮可以直接使用 Button 支持的各种属性。

在 Android 中，可使用两种方法向屏幕中添加复选按钮，一种是通过在 XML 布局文件中使用<CheckBox>标记添加，另一种是在 Java 文件中通过 new 关键字创建。建议使用第一种方法，也就是通过<CheckBox>在 XML 布局文件中添加。在 XML 布局文件中添加复选按钮的格式为：

```
<CheckBox
    android:id = "@ + id/ID 号"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_alignLeft = "@ + id/textView1"
    android:layout_below = "@ + id/textView1"
    android:layout_marginLeft = "36dp"
    android:layout_marginTop = "68dp"
    android:text = "显示文本" />
```

由于复选按钮可以选中多项，所以为了确定用户是否选择了某一项，还需要为每一个选项添加事件监听器。

下面通过一个实例来演示 CheckBox 控件的用法。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 CheckBox_test。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中声明 3 个 CheckBox 控件、一个 EditText 控件及一个 Button 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
    <EditText
        android:id = "@ + id/EditText1"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:singleLine = "false">
    </EditText>
    <CheckBox
        android:text = "跳舞"
        android:id = "@ + id/CheckBox1"
        android:textSize = "35dip"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content">
    </CheckBox>
    <CheckBox
        android:text = "游泳"
        android:id = "@ + id/CheckBox2"
        android:textSize = "35dip"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content">
    </CheckBox>
```

```
        <CheckBox
            android:text="爬山"
            android:id="@+id/CheckBox3"
            android:textSize="35dip"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
        </CheckBox>
        <Button
            android:text="确定"
            android:id="@+id/Button1"
            android:textSize="35dip"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
        </Button>
    </LinearLayout>
```

(3) 打开 src\checkbox_test 包下 MainActivity.java 文件，在文件中实现爱好的选择，并把对应的选择显示在 TextView 控件中。代码为：

```
package fs.checkbox_test;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
public class MainActivity extends Activity
{
    String result;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final CheckBox cb1 = (CheckBox)this.findViewById(R.id.CheckBox1);
        final CheckBox cb2 = (CheckBox)this.findViewById(R.id.CheckBox2);
        final CheckBox cb3 = (CheckBox)this.findViewById(R.id.CheckBox3);
        Button but = (Button)this.findViewById(R.id.Button1);
        final EditText et = (EditText)this.findViewById(R.id.EditText1);
        //添加事件监听器
        but.setOnClickListener
        (
                new OnClickListener()
                {
                    public void onClick(View v)
                    {
                        result = "选择为：";
                        et.setText("");
                        if(cb1.isChecked())
                        {
```

```
                    result += "唱歌  ";
                }if(cb2.isChecked())
                {
                    result += "游泳  ";
                }if(cb3.isChecked())
                {
                    result += "写 Java 程序\n";
                }
                et.setText(result.toString().trim());
            }
        }
    );
}
}
```

运行程序,效果如图 3-10 所示。



图 3-10　复选按钮实例

3.3.5　CheckedTextView 控件

在 Android 技术中实现选中的 checked 效果其实还有另外一个控件也可以实现,即 CheckedTextView 控件。

类 CheckedTextView 继承超类 TextView 并实现 Checkable 接口。当 ListView 的 setChoiceMode 方法并设定为 CHOICE_MODE_SINGLE 或者 CHOICE_MODE_MULTIPLE,而非 CHOICE_MODE_NONE 时,使用此类是很有用的。

在 XML 布局文件中添加 CheckedTextView 控件的格式为:

```
<CheckedTextView
        android:id = "@ + id/checkedTextView1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/textView1"
        android:layout_below = "@ + id/textView1"
        android:layout_marginLeft = "42dp"
        android:layout_marginTop = "136dp"
        android:text = "CheckedTextView" />
```

下面通过一个实例来演示 CheckedTextView 控件的用法。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 CheckedTextView_test。

(2) 打开 res\layout 目录下的 main. xml 文件，代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<ScrollView xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@ + id/scrollView1"
    android:layout_width = "match_parent"
    android:layout_height = "wrap_content">
    <LinearLayout
        android:padding = "10px"
        android:orientation = "vertical"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:background = "#aabbcc">
        <CheckedTextView
            android:tag = "a1"
            android:id = "@ + id/checkedTextView1"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:checkMark = "?android:attr/listChoiceIndicatorMultiple"
            android:text = "checkedTextView1" />
        <CheckedTextView
            android:tag = "a2"
            android:id = "@ + id/checkedTextView2"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:checkMark = "?android:attr/listChoiceIndicatorMultiple"
            android:text = "checkedTextView2" />
        <CheckedTextView
            android:tag = "a3"
            android:id = "@ + id/checkedTextView3"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:checkMark = "?android:attr/listChoiceIndicatorMultiple"
            android:text = "checkedTextView4" />
        <CheckedTextView
            android:tag = "a4"
            android:id = "@ + id/checkedTextView4"
            android:layout_width = "fill_parent"
```

```
        android:layout_height = "wrap_content"
        android:checkMark = "?android:attr/listChoiceIndicatorMultiple"
        android:text = "checkedTextView5" />
    < CheckedTextView
        android:tag = "a5"
        android:id = "@ + id/checkedTextView5"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:checkMark = "?android:attr/listChoiceIndicatorMultiple"
        android:text = "checkedTextView6" />
    < Button
        android:text = "Button"
        android:id = "@ + id/button1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"/>
    < CheckedTextView
        android:tag = "A"
        android:id = "@ + id/checkedTextViewa"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:checkMark = "?android:attr/listChoiceIndicatorSingle"
        android:text = "checkedTextViewa" />
    < CheckedTextView
        android:tag = "B"
        android:id = "@ + id/checkedTextViewb"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:checkMark = "?android:attr/listChoiceIndicatorSingle"
        android:text = "checkedTextViewb" />
    < CheckedTextView
        android:tag = "C"
        android:id = "@ + id/checkedTextViewc"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:checkMark = "?android:attr/listChoiceIndicatorSingle"
        android:text = "checkedTextViewc" />
    < CheckedTextView
        android:tag = "D"
        android:id = "@ + id/checkedTextViewd"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:checkMark = "?android:attr/listChoiceIndicatorSingle"
        android:text = "checkedTextViewd" />
    < Button
        android:text = "Button"
        android:id = "@ + id/button2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"/>
  </LinearLayout >
</ScrollView >
```

(3) 打开 src\fs. checkedtextview_test 包下的 MainActivity. java 文件，在文件中实现单选按钮及多选按钮效果。代码为：

```
package fs.checkedtextview;
import java.util.ArrayList;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.CheckedTextView;
public class MainActivity extends Activity {
    private CheckedTextView checkedTextViewMul1;
    private CheckedTextView checkedTextViewMul2;
    private CheckedTextView checkedTextViewMul3;
    private CheckedTextView checkedTextViewMul4;
    private CheckedTextView checkedTextViewMul5;
    private CheckedTextView checkedTextViewSinglea;
    private CheckedTextView checkedTextViewSingleb;
    private CheckedTextView checkedTextViewSinglec;
    private CheckedTextView checkedTextViewSingled;
    private Button getMulCheckedTextValue;
    private Button getSingleCheckedTextValue;
    private ArrayList<Integer> mulCheckedTextViewIdArray = new ArrayList();
    private ArrayList<Integer> singleCheckedTextViewIdArray = new ArrayList();
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        getMulCheckedTextValue = (Button) this.findViewById(R.id.button1);
        getSingleCheckedTextValue = (Button) this.findViewById(R.id.button2);
        checkedTextViewMul1 = (CheckedTextView) this
                .findViewById(R.id.checkedTextView1);
        checkedTextViewMul1.setChecked(true);
        checkedTextViewMul2 = (CheckedTextView) this
                .findViewById(R.id.checkedTextView2);
        checkedTextViewMul3 = (CheckedTextView) this
                .findViewById(R.id.checkedTextView3);
        checkedTextViewMul3.setChecked(true);
        checkedTextViewMul4 = (CheckedTextView) this
                .findViewById(R.id.checkedTextView4);
        checkedTextViewMul5 = (CheckedTextView) this
                .findViewById(R.id.checkedTextView5);
        checkedTextViewMul5.setChecked(true);
        mulCheckedTextViewIdArray.add(checkedTextViewMul1.getId());
        mulCheckedTextViewIdArray.add(checkedTextViewMul2.getId());
        mulCheckedTextViewIdArray.add(checkedTextViewMul3.getId());
        mulCheckedTextViewIdArray.add(checkedTextViewMul4.getId());
        mulCheckedTextViewIdArray.add(checkedTextViewMul5.getId());
```

```
OnClickListener checkedTextViewMulListenerRef = new OnClickListener() {
    public void onClick(View arg0) {
        ((CheckedTextView) arg0).toggle();
    }
};
checkedTextViewMul1.setOnClickListener(checkedTextViewMulListenerRef);
checkedTextViewMul2.setOnClickListener(checkedTextViewMulListenerRef);
checkedTextViewMul3.setOnClickListener(checkedTextViewMulListenerRef);
checkedTextViewMul4.setOnClickListener(checkedTextViewMulListenerRef);
checkedTextViewMul5.setOnClickListener(checkedTextViewMulListenerRef);
getMulCheckedTextValue.setOnClickListener(new OnClickListener() {
    public void onClick(View arg0) {
        for (int i = 0; i < mulCheckedTextViewIdArray.size(); i++) {
CheckedTextView findCheckedTextViewRef = (CheckedTextView) MainActivity.this
                            .findViewById(mulCheckedTextViewIdArray.get(i));
            if (findCheckedTextViewRef.isChecked() == true) {
                Log.v("选中的 checkbox 值是",""
                        + findCheckedTextViewRef.getTag());
            }
        }
    }
});
checkedTextViewSinglea = (CheckedTextView) this
        .findViewById(R.id.checkedTextViewa);
checkedTextViewSingleb = (CheckedTextView) this
        .findViewById(R.id.checkedTextViewb);
checkedTextViewSinglec = (CheckedTextView) this
        .findViewById(R.id.checkedTextViewc);
checkedTextViewSingled = (CheckedTextView) this
        .findViewById(R.id.checkedTextViewd);
singleCheckedTextViewIdArray.add(checkedTextViewSinglea.getId());
singleCheckedTextViewIdArray.add(checkedTextViewSingleb.getId());
singleCheckedTextViewIdArray.add(checkedTextViewSinglec.getId());
singleCheckedTextViewIdArray.add(checkedTextViewSingled.getId());
OnClickListener checkedTextViewSinglelListenerRef = new OnClickListener() {
    public void onClick(View arg0) {
        for (int i = 0; i < singleCheckedTextViewIdArray.size(); i++) {
if (singleCheckedTextViewIdArray.get(i).intValue() != ((CheckedTextView) arg0)
                            .getId()) {
                ((CheckedTextView) MainActivity.this
                        .findViewById(singleCheckedTextViewIdArray
                                .get(i))).setChecked(false);
            } else {
                ((CheckedTextView) MainActivity.this
                        .findViewById(singleCheckedTextViewIdArray
                                .get(i))).setChecked(true);
            }
        }
    }
};
checkedTextViewSinglea
```

```
                .setOnClickListener(checkedTextViewSinglelListenerRef);
        checkedTextViewSingleb
                .setOnClickListener(checkedTextViewSinglelListenerRef);
        checkedTextViewSinglec
                .setOnClickListener(checkedTextViewSinglelListenerRef);
        checkedTextViewSingled
                .setOnClickListener(checkedTextViewSinglelListenerRef);
        getSingleCheckedTextValue.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
                for (int i = 0; i < singleCheckedTextViewIdArray.size(); i++) {
                    CheckedTextView eachCheckedTextViewRef = ((CheckedTextView)
MainActivity.this.findViewById(singleCheckedTextViewIdArray.get(i)));
                    if (eachCheckedTextViewRef.isChecked() == true) {
                        Log.v("单选选中了:",""
                                + eachCheckedTextViewRef.getTag().toString());
                    }
                }
            }
        });
    }
}
```

运行程序,效果如图 3-11 所示。

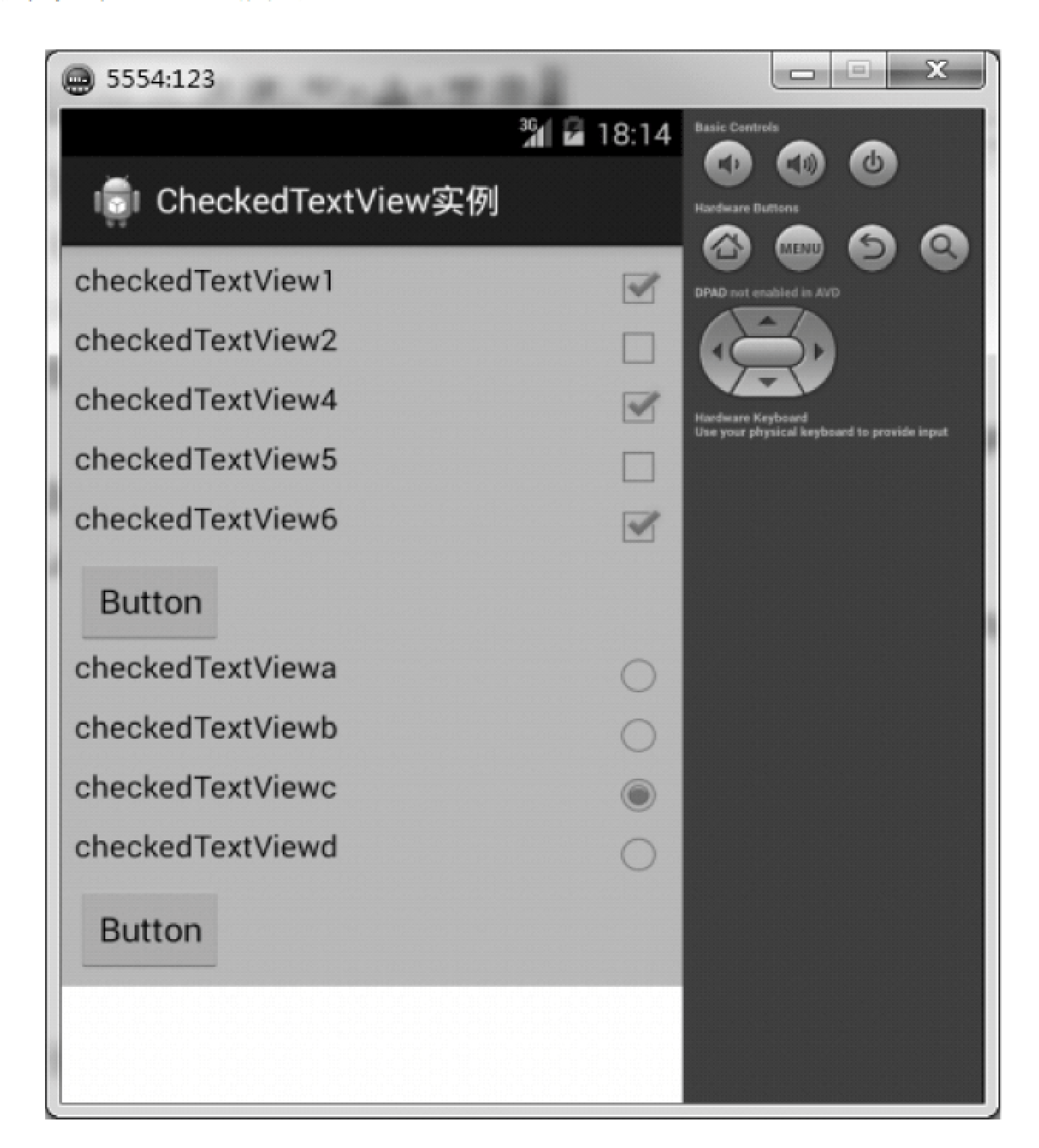


图 3-11　CheckedTextView 控件效果

3.3.6 Android 开关按钮

ToggleButton(开关按钮)的继承关系如图 3-12 所示。ToggleButton 的状态只能是选中和未选中状态,并且需要为不同的状态设置不同的显示文本。除了继承自父类的一些属性和方法外,ToggleButton 也具有一些自己的 ToggleButton 属性,如表 3-4 所示。

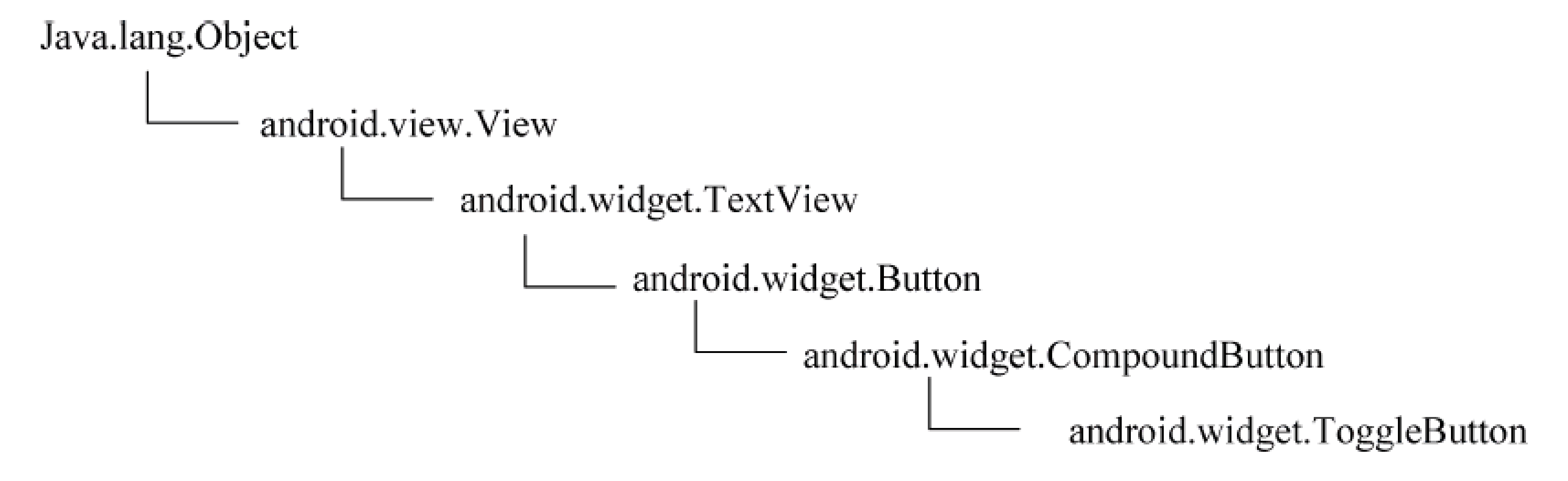


图 3-12 双按钮的继承关系

表 3-4 ToggleButton 支持的 XML 属性及描述

XML 属性	方 法	描 述
android:checked	setChecked(Boolean)	设置该按钮是否被选中
android:textOff		设置当该按钮没有被选中时显示的文本
android:textOn		设置当该按钮被选中时显示的文本

下面通过一个实例来演示 ToggleButton 控件的用法。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 ToggleButton_test。

(2) 打开 res\values 目录下的 strings.xml 文件,为声明变量并赋值,代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < string name = "app_name"> ToggleButton 实例</string >
    < string name = "action_settings"> Settings </string >
    < string name = "hello_world"> Hello world!</string >
    < string name = "on">开灯</string >
    < string name = "off">关灯</string >
</resources >
```

(3) 打开 res\layout 目录下的 main.xml 布局文件,在文件中声明一个 ImageView 控件及一个 ToggleButton 控件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "#aabbcc">
<! -- 声明一个 ImageView 控件 -- >
    < ImageView
        android:id = "@ + id/ImageView1"
        android:layout_width = "wrap_content"
```

```
            android:layout_height = "wrap_content"
            android:layout_gravity = "center_horizontal"
            android:src = "@drawable/right1" />
<! -- 声明一个 ToggleButton 控件 -->
    <ToggleButton
            android:id = "@ + id/ToggleButton1"
            android:layout_width = "140dip"
            android:layout_height = "wrap_content"
            android:layout_gravity = "center_horizontal"
            android:textOff = "@string/on"
            android:textOn = "@string/off" />
</LinearLayout>
```

(4) 打开 src\fs. ToggleButton_test 包下的 MainActivity. java 文件，在文件中实现灯泡的开与关功能。代码为：

```
package fs.togglebutton_test;
import android.app.Activity;
import android.os.Bundle;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.ImageView;
import android.widget.ToggleButton;
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ToggleButton tb = (ToggleButton) this.findViewById(R.id.ToggleButton1);
        tb.setOnCheckedChangeListener(new OnCheckedChangeListener() {
          //为 ToggleButton 添加监听器
            @Override
            public void onCheckedChanged(CompoundButton buttonView,
                    boolean isChecked) {            //重写 onCheckedChanged 方法
                setBulbState(isChecked);            //设置控件状态
            }
        });
    }
    //方法：设置程序状态的方法
    public void setBulbState(boolean state) {
        //设置图片状态
        ImageView iv = (ImageView) findViewById(R.id.ImageView1);
    iv.setImageResource((state) ? R.drawable.right2 : R.drawable.right1);  //设置图片资源
        //设置 ToggleButton 状态
        ToggleButton tb = (ToggleButton) this.findViewById(R.id.ToggleButton1);
        tb.setChecked(state);                       //设置 ToggleButton 选中状态
    }
}
```

运行程序，效果如图 3-13 所示。

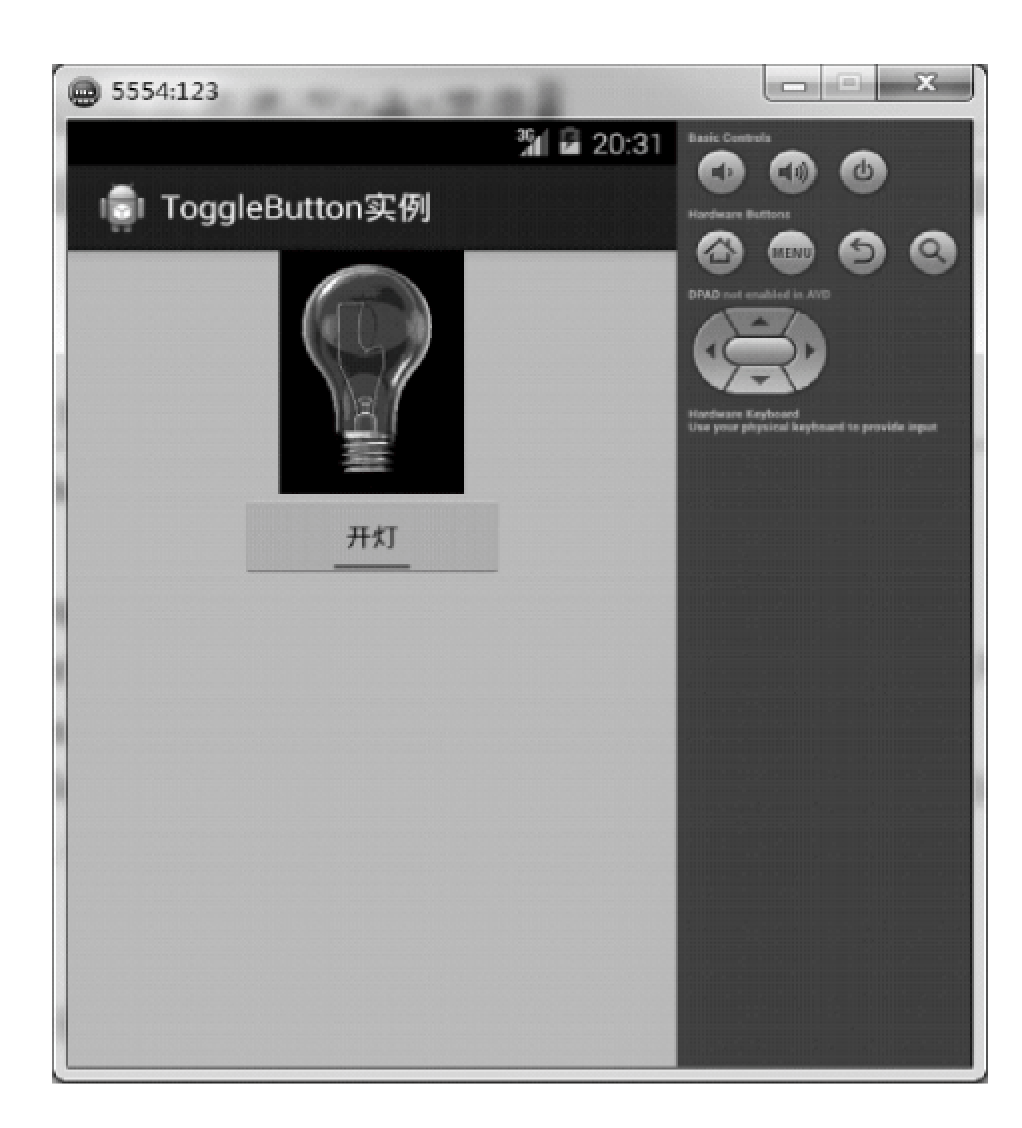




图 3-13　ToggleButton 控件效果

3.4　Android 列表类控件

在 Android 中，提供了两种列表类控件，一种是列表选择框，通常用于实现类似于网页中常见的下拉列表框；另一种是列表视图，通常用于实现在一个窗口中只显示一个列表。下面给予介绍。

3.4.1　Android 列表选择框

Android 中提供的列表选择框(Spinner)相当于在网页中常见的下拉列表框，通常用于提供一系列可选择的列表项，供用户进行选择，从而方便用户。

Spinner 位于 android. widget 包下，它每次只显示用户选中的元素，当用户再次单击时，会弹出选择列表供用户选择，而选择列表的元素同样来自适配器，如图 3-14 所示为该类的继承树。

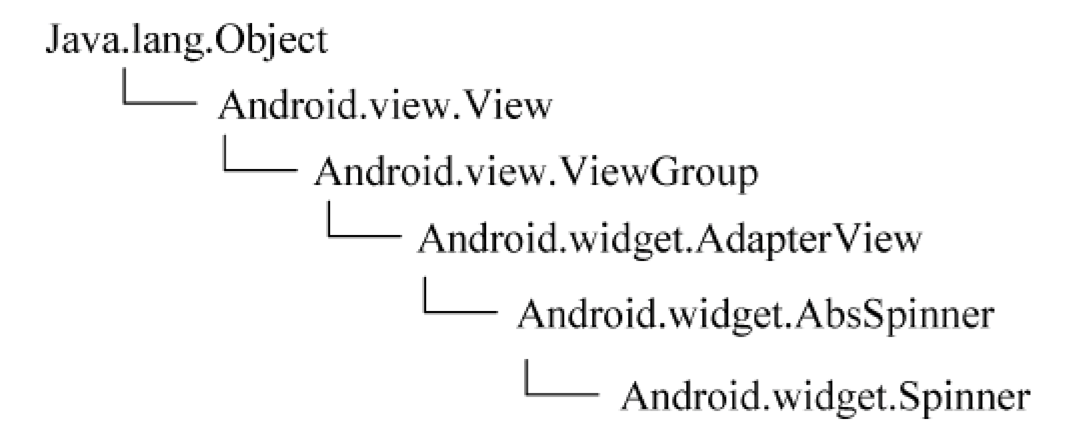


图 3-14　Spinner 类的继承树

在 Android 中，可以使用两种方法向屏幕中添加列表选择框，一种是通过在 XML 布局文件中使用<Spinner>在 XML 布局文件中添加。在 XML 布局文件中添加列表选择框的格式为：

```
<Spinner
```

```
    android:prompt = "@string/info"
    android:id = "@ + id/ID 号 "
    android:entries = "@array/数组名称"
    android:layout_width = "match_parent"
    android:layout_height = "wrap_content" />
```

其中，android:entries 为可选属性，用于指定列表项，如果不在布局文件中指定该属性，可以在 Java 代码中通过指定适配器的方式指定；android:prompt 属性也是可选属性，用于指定列表选择框的标题。

在通常情况下，如果列表选择框中要显示的列表项是可知的，那么将其保存在数组资源文件中，然后通过数组资源来为列表选择框指定列表项。这样，就可以在不编写 Java 代码的情况下实现一个列表选择框。

下面通过一个实例来演示 Spinner 控件的用法。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Spinner_test。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中声明 TextView 控件和一个 Spinner 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
android:orientation = "vertical"
android:layout_width = "fill_parent"
android:layout_height = "fill_parent"
android:background = " # aabbcc" >
< TextView
    android:text = "@string/ys"
    android:id = "@ + id/TextView1"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:textSize = "28dip"/>
< Spinner
    android:id = "@ + id/Spinner1"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"/>
</LinearLayout >
```

(3) 打开 res\values 目录下的 string.xml 文件，为变量赋值，代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < string name = "app_name"> Spinner 案例</string >
    < string name = "action_settings"> Settings </string >
    < string name = "hello_world"> Hello world! </string >
    < string name = "ys">您的爱好</string >
    < string name = "lq">篮球</string >
    < string name = "zp">足球</string >
    < string name = "pq">排球</string >
</resources >
```

(4) 在 res\values 目录下创建一个颜色资源文件 color.xml，代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
```

```
<resources>
<color name = "red">#fd8d8d</color>
<color name = "green">#9cfda3</color>
<color name = "blue">#8d9dfd</color>
<color name = "white">#FFFFFF</color>
<color name = "black">#000000</color>
</resources>
```

(5) 打开 src\fs.spinner_test 包下的 MainActivity.java 文件，在文件中实现列表框的选择，代码为：

```
package fs.spinner_test;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.AdapterView.OnItemSelectedListener;
public class MainActivity extends Activity {
    final static int WRAP_CONETNT = -2;               //表示 WRAP_CONTENT 的常量
    //所有资源的图片(足球、篮球、排球) id 的数组
    int[] drawableIds = { R.drawable.fb1,R.drawable.fb2,R.drawable.fb3 };
    //所有资源字符串 (足球、篮球、排球) id 的数组
    int[] msgIds = { R.string.zp,R.string.lq,R.string.pq };
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Spinner sp = (Spinner) findViewById(R.id.Spinner1);
        BaseAdapter ba = new BaseAdapter() {
            public int getCount() {
                //一共 3 个选项
                return 3;
                }
            public Object getItem(int position) {
                return null;
                }
            public long getItemId(int position) {
                return 0;
                }
            @SuppressLint("ResourceAsColor")
            public View getView(int position,View convertView,ViewGroup parent) {
            //动态生成每个下拉项对应的 View,每个下拉项 View 由 LinearLayout
                //中包含一个 ImageView 及一个 TextView 构成
                //初始化 LinearLayout
                LinearLayout b = new LinearLayout(MainActivity.this);
                b.setOrientation(LinearLayout.HORIZONTAL);
                //初始化 ImageView
```

```
                    ImageView ii = new ImageView(MainActivity.this);
                    ii.setImageDrawable((getResources().getDrawable(drawableIds[position])));
                    b.addView(ii);
                    //初始化 TextView
                    TextView tv = new TextView(MainActivity.this);
                    tv.setText(" " + getResources().getText(msgIds[position]));
                    tv.setTextColor(R.color.black);
                    tv.setTextSize(24);
                    b.addView(tv);
                    return b;
                    }
                };
                //为 Spinner 设置内容适配器
                sp.setAdapter(ba);
                sp.setOnItemSelectedListener(new OnItemSelectedListener() {
        public void onItemSelected(AdapterView<?> parent,View view,int position,long id) {
                    //获取主界面 TextView
                    TextView tv = (TextView) findViewById(R.id.TextView1);
                    //获取当前选中选项对应的 LinearLayout
                    LinearLayout ll = (LinearLayout) view;
                    //获取其中的 TextView
                    TextView tvn = (TextView) ll.getChildAt(1);
                    //用 StringBuilder 动态生成信息
                    StringBuilder sb = new StringBuilder();
                    sb.append(getResources().getText(R.string.ys));
                    sb.append(":");
                    sb.append(tvn.getText());
                    //信息设置进住界面
                    tv.setText(sb.toString());
                    }
                public void onNothingSelected(AdapterView<?> parent) {}
                });
            }
    }
```

运行程序,默认界面如图 3-15(a)所示,当单击列表框右侧的"三角"符号,弹出列表选择项,效果如图 3-15(b)所示,选择对应的喜好,即在文本框中显示对应的选项,如图 3-15(c)所示。




(a) 默认界面



(b) 弹出列表选项



(c) 选择选项

图 3-15　下拉列表控件效果

3.4.2 Android 文本列表框

在 Android 中提供了 ListView 实现文本列表框，其有几种功能，下面分别给予介绍。

1. 显示文本列表功能

在 3.4.1 节使用了 Spinner 控件来进行弹出列表框选择其中条目的示例，其实在 Android 中还有一个重量级的列表对象，它就是 ListView。在 Android 中的 ListView 的使用率基本达到 90%，所以掌握 ListView 对象的使用是掌握 Android 处理数据和展示数据的基本技能。

单独使用 ListView 控件可以不弹出对话框来进行列表选项的选择，因为整个界面只存在一个 ListView 控件。

下面通过一个实例来演示在 ListView 控件中显示文本列表功能。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 ListView_test1。

(2) 打开 res\layout 目录下的 main.xml 文件，在布局文件中定义一个 TextView 控件及一个 ListView 控件。代码为：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#aabbcc">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="城市列表" />
    <ListView
        android:layout_height="wrap_content"
        android:id="@+id/listView1"
        android:layout_width="match_parent"/>
</LinearLayout>
```

(3) 打开 src\fs.listview_test1 包下的 MainActivity.java 文件，用于实现 ListViewe 的文本列表功能。代码为：

```
package fs.listview_test1;
import java.util.ArrayList;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.AdapterView.OnItemClickListener;
public class MainActivity extends Activity {
    private ListView listView1;
    private ArrayList cityList = new ArrayList();
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        listView1 = (ListView) this.findViewById(R.id.listView1);
        cityList.add("北京");
        cityList.add("上海");
        cityList.add("天津");
        cityList.add("南京");
        ArrayAdapter arrayAdapterRef = new ArrayAdapter(this,
                android.R.layout.simple_list_item_1,cityList);
        listView1.setAdapter(arrayAdapterRef);
        listView1.setOnItemClickListener(new OnItemClickListener() {
            public void onItemClick(AdapterView<?> arg0,View arg1,int arg2,
                    long arg3) {
                Log.v("listView1.setOnItemClickListener",""
                        + cityList.get(arg2));
            }
        });
    }
}
```

运行程序,效果如图 3-16 所示。



图 3-16　ListView 的文本列表功能

2. 使用多选 Checkedbox 控件

下面通过一个实例来演示在 ListView 控件中使用多选 Checkedbox 控件及有全选、反选和取值的功能。其具体操作步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 ListView_test2。

(2) 打开 res\layout 目录下的 main.xml 文件,在文件中声明一个 TextView 控件、一个 ListView 控件及 3 个 Button 控件。代码为:

```xml
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
    <TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "水果名称" />
    <ListView
        android:layout_weight = "1"
        android:id = "@ + id/listView1"
        android:layout_height = "wrap_content"
        android:layout_width = "match_parent"/>
    <LinearLayout
        android:gravity = "center"
        android:orientation = "horizontal"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content">
        <Button
            android:text = "全选"
            android:id = "@ + id/button1"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"/>
        <Button
            android:text = "反选"
            android:id = "@ + id/button2"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"/>
        <Button
            android:text = "取值"
            android:id = "@ + id/button3"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"/>
    </LinearLayout>
</LinearLayout>
```

(3) 打开 src\fs.listview_test2 包下的 MainActivity.java，在该文件中实现多选、全选、反选及取值功能等。代码为：

```java
package fs.listview_test2;
import java.util.ArrayList;
import java.util.List;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.util.SparseBooleanArray;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.AdapterView.OnItemClickListener;
```

```
public class MainActivity extends Activity {
    private List cityList = new ArrayList();
    private ListView listView;
    private Button button1;                             //全选
    private Button button2;                             //反选
    private Button button3;                             //取值
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        listView = (ListView) this.findViewById(R.id.listView1);
        button1 = (Button) this.findViewById(R.id.button1);
        button2 = (Button) this.findViewById(R.id.button2);
        button3 = (Button) this.findViewById(R.id.button3);
        final boolean[] isCheckedArray = new boolean[8];
        isCheckedArray[0] = false;
        isCheckedArray[1] = true;                       //默认值为 true
        isCheckedArray[2] = false;
        isCheckedArray[3] = true;                       //默认值为 true
        isCheckedArray[4] = false;
        isCheckedArray[5] = true;                       //默认值为 true
        isCheckedArray[6] = false;
        isCheckedArray[7] = true;                       //默认值为 true
        cityList.add("苹果");
        cityList.add("香蕉");
        cityList.add("雪梨");
        cityList.add("芒果");
        cityList.add("桃子");
        cityList.add("葡萄");
        cityList.add("甘蔗");
        cityList.add("西瓜");
        ArrayAdapter adapter = new ArrayAdapter(this,
                android.R.layout.simple_list_item_multiple_choice,cityList);
        listView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
        listView.setAdapter(adapter);
        listView.setOnItemClickListener(new OnItemClickListener() {
            public void onItemClick(AdapterView<?> arg0,View arg1,int arg2,
                    long arg3) {
                Log.v("------------","" + ((TextView) arg1).getText());
            }
        });
        //赋初始值
        for (int i = 0; i < isCheckedArray.length; i++) {
            listView.setItemChecked(i,isCheckedArray[i]);
        }
        //全选
        button1.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
                Log.v("单击了全选","单击了全选");
                for (int i = 0; i < isCheckedArray.length; i++) {
                    listView.setItemChecked(i,true);
                }
            }
        });
        //反选
        button2.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
```

```
                Log.v("单击了反选","单击了反选");
                SparseBooleanArray sparseBooleanArrayRef = listView
                        .getCheckedItemPositions();
                for (int i = 0; i < sparseBooleanArrayRef.size(); i++) {
                    if (sparseBooleanArrayRef.get(i) == true) {
                        listView.setItemChecked(i,false);
                    } else {
                        listView.setItemChecked(i,true);
                    }
                }
            }
        });
        //取值
        button3.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
                Log.v("单击了取值","单击了取值");
                SparseBooleanArray sparseBooleanArrayRef = listView
                        .getCheckedItemPositions();
                for (int i = 0; i < sparseBooleanArrayRef.size(); i++) {
                    if (sparseBooleanArrayRef.get(i) == true) {
                        Log.v("值为: ","" + listView.getAdapter().getItemId(i)
                                + " " + listView.getAdapter().getItem(i));
                    }
                }
            }
        });
    }
}
```

运行程序,效果如图 3-17 所示。



图 3-17　ListView 控件实现多选功能效果

3. 使用单选 RadioButton 控件

前面的演示是在 ListView 控件中显示多选 Checkedbox 控件来进行全选、反选和取值的实例。其中在 ListView 中的每一条目还可以是 RadioButton 类型的控件。

下面通过一个实例来演示在 ListView 中实现单选。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 ListView_test3。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明一个 TextView 控件及一个 ListView 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = " # aabbcc">
    < TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "水果名称" />
    < ListView
        android:layout_weight = "1"
        android:id = "@ + id/listView1"
        android:layout_height = "wrap_content"
        android:layout_width = "match_parent"/>
</LinearLayout >
```

(3) 打开 src\fs.listview_test3 包下的 MainActivity.java 文件，在文件中实现单选功能。代码为：

```
package fs.listview_test3;
import java.util.ArrayList;
import java.util.List;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.AdapterView.OnItemClickListener;
public class MainActivity extends Activity {
    private List cityList  =  new ArrayList();
    private ListView listView;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //显示列表的用户名
        listView  =  (ListView) this.findViewById(R.id.listView1);
        final boolean[] isCheckedArray  =  new boolean[8];
        isCheckedArray[0]  =  false;
        isCheckedArray[1]  =  true;
```

```
                isCheckedArray[2] = false;
                isCheckedArray[3] = false;
                isCheckedArray[4] = false;
                isCheckedArray[5] = false;
                isCheckedArray[6] = false;
                isCheckedArray[7] = false;
                cityList.add("苹果");
                cityList.add("香蕉");
                cityList.add("雪梨");
                cityList.add("芒果");
                cityList.add("桃子");
                cityList.add("葡萄");
                cityList.add("甘蔗");
                cityList.add("西瓜");
                ArrayAdapter adapter = new ArrayAdapter(this,
                        android.R.layout.simple_list_item_single_choice,cityList);
                listView.setChoiceMode(ListView.CHOICE_MODE_SINGLE);
                listView.setAdapter(adapter);
                listView.setOnItemClickListener(new OnItemClickListener() {
                    public void onItemClick(AdapterView<?> arg0,View arg1,int arg2,
                            long arg3) {
                        Log.v("------------","" + ((TextView) arg1).getText());
                    }
                });
                //设默认选中状态
                for (int i = 0; i < isCheckedArray.length; i++) {
                    listView.setItemChecked(i,isCheckedArray[i]);
                }
            }
        }
```

运行程序，效果如图 3-18 所示。



图 3-18　ListView 中实现单选功能效果

4. 添加及删除条目

除了在 ListView 中实现文本列表功能、多选功能、单选功能外，还可以在 ListView 中添加及删除条目。下面通过一个实例来演示在 ListView 中添加及删除条目，其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 ListView_test4。

(2) 打开 res\layout 目录下的 main.xml 布局文件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
<LinearLayout
        android:orientation = "vertical"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content">
        <Button
            android:id = "@ + id/button1"
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
            android:text = "删除" />
        <Button
            android:id = "@ + id/button2"
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
            android:text = "添加" />
    </LinearLayout>
    <LinearLayout
        android:orientation = "vertical"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent">
<TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "水果条目列表" />
        <ListView
            android:id = "@ + id/listView1"
            android:layout_height = "wrap_content"
            android:layout_width = "match_parent"/>
    </LinearLayout>
</LinearLayout>
```

(3) 打开 src\fs.listview_test4 包下的 MainActivity.java 文件，在文件中实现条件添加与删除。代码为：

```
package fs.listview_test4;
import java.util.ArrayList;
import java.util.List;
```

```
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
public class MainActivity extends Activity {
    private Button button1;
    private Button button2;
    private ListView listView1;
    final private List dataList = new ArrayList();
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        listView1 = (ListView) this.findViewById(R.id.listView1);
        button1 = (Button) this.findViewById(R.id.button1);
        button2 = (Button) this.findViewById(R.id.button2);
        dataList.add("我是苹果,序号为: 1");
        dataList.add("我是香蕉,序号为: 2");
        dataList.add("我是雪梨,序号为: 3");
        dataList.add("我是西瓜,序号为: 4");
        dataList.add("我是葡萄,序号为: 5");
        dataList.add("我是甘蔗,序号为: 6");
        dataList.add("我是橘子,序号为: 7");
        dataList.add("我是桃子,序号为: 8");
        dataList.add("我是杨桃,序号为: 9");
        dataList.add("我是石榴,序号为: 10");
        final ArrayAdapter adapter = new ArrayAdapter(this,
                android.R.layout.simple_list_item_1,dataList);
        listView1.setAdapter(adapter);
        button1.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
                dataList.add(0,"  ");
                adapter.notifyDataSetChanged();
                Log.v("!","dataList size() = " + dataList.size());
            }
        });
        button2.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
                dataList.remove(0);
                adapter.notifyDataSetChanged();
                Log.v("!","dataList size() = " + dataList.size());
            }
        });
    }
}
```

运行程序,效果如图 3-19 所示。







图 3-19　条目的删除及添加

3.5　Android 条类控件

在 Android 中也提供了相关控件实现进度条、滚动条、拖动条以及星级评分条等功能。下面给予介绍。

3.5.1　Android 进度条

当一个应用程序在后台执行时，前台界面不会有任何信息，这时用户根本不知道程序是否在执行或执行进度等，因此需要使用进度条来提示程序执行的进度。在 Android 中，进度条使用 ProgressBar 表示，用于向用户显示某个耗时操作完成的百分比。

在屏幕中添加进度条，可以在 XML 布局文件中通过<ProgressBar>标记实现，其语法格式为：

```
<!-- 圆形进度条 -->
    <ProgressBar
        android:id = "@ + id/progressBar1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_below = "@ + id/textView1"
        android:layout_centerHorizontal = "true"
        android:layout_marginTop = "24dp" />
<!-- 长形进度条 -->
    <ProgressBar
        android:id = "@ + id/progressBar2"
        style = "?android:attr/progressBarStyleHorizontal"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignRight = "@ + id/progressBar1"
        android:layout_below = "@ + id/progressBar1"
        android:layout_marginTop = "52dp" />
```

Android 当中的进度条 ProgressBar 有两种进度条，一种为垂直(圆圈)，一种为水平(水平线)。

Android 支持几种风格的进度条，通过 style 属性可以为 ProgressBar 指定风格。该属性可支持以下几个属性值。

- @android:style/Widget.ProgressBar.Horizontal：水平进度条。
- @android:style/Widget.ProgressBar.Inverse：普通大小进度条。
- @android:style/Widget.ProgressBar.Large：大进度条。
- @android:style/Widget.ProgressBar.Large.Inverse：普通大进度条。
- @android:style/Widget.ProgressBar.Small：小进度条。
- @android:style/Widget.ProgressBar.Small.Inverse：普通小进度条。

除此之外，ProgressBar 还支持如表 3-5 所示的常用 XML 属性。

表 3-5 ProgressBar 常用的 XML 属性

XML 属性	描　　述
android:max	设置该进度条的最大值
android:progress	设置该进度条的已完成进度值
android:progressDrawable	设置该进度条的轨道的绘制形式
android:indeterminate	该属性设为 true，设置进度条不精确显示进度
android:indeterminateDrawable	设置绘制不显示进度的进度条的 Drawable 对象
android:indeterminateDuration	设置不精确显示进度的持续时间

表 3-5 中 android:progressDrawable 用于指定进度条的轨道的绘制形式，该属性可指定为一个 LayerDrawable 对象（该对象可通过在 XML 文件中用<layer-list>元素进行配置）的引用。

ProgressBar 提供了如下方法来操作进度。

- setProgress(int)：设置进度的完成百分比。
- incrementProgressBy(int)：设置进度条的进度增加或减少。当参数为正数时进度增加；当参数为负数时进度减少。

下面通过一个实例来演示进度条的用法。该实例实现在某些操作的进度中的可视指示器，为用户呈现操作的进度，它还有一个次要的进度条，用来显示中间进度，例如在流媒体播放的缓冲区的进度。一个进度条也可不确定其进度。在不确定模式下，进度条显示循环动画。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 ProgressBar_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件，代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "match_parent"
    android:layout_height = "wrap_content"
    android:background = "#aabbcc">
    < ProgressBar android:id = "@ + id/progress_horizontal"
        style = "?android:attr/progressBarStyleHorizontal"
        android:layout_width = "200dip"
        android:layout_height = "wrap_content"
        android:max = "100"
```

```
            android:progress = "50"
            android:secondaryProgress = "75" />
        < TextView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "默认进度条" />
        < LinearLayout
            android:orientation = "horizontal"
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content">
            < Button android:id = "@ + id/decrease"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:text = "减少" />
            < Button android:id = "@ + id/increase"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:text = "增加" />
        </LinearLayout >
        < TextView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "自定义进度条" />
        < LinearLayout
            android:orientation = "horizontal"
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content">
            < Button android:id = "@ + id/decrease_secondary"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:text = "第二减少" />
            < Button android:id = "@ + id/increase_secondary"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:text = "第二增加" />
        </LinearLayout >
    </LinearLayout >
```

(3) 打开 src\fs. progressbar 包下的 MainActivity. java 文件,在文件中实现主进度条与次进度的增加与减小。代码为:

```
package fs.progressbar_test;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.widget.Button;
import android.widget.ProgressBar;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //TODO 自动存根法
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_PROGRESS);
        setContentView(R.layout.main);
        setProgressBarVisibility(true);
```

```
            final ProgressBar progressHorizontal = (ProgressBar) findViewById(R.id.progress_
horizontal);
            setProgress(progressHorizontal.getProgress() * 100);
            setSecondaryProgress(progressHorizontal.getSecondaryProgress() * 100);
            Button button = (Button) findViewById(R.id.increase);
            button.setOnClickListener(new Button.OnClickListener() {
                public void onClick(View v) {
                    progressHorizontal.incrementProgressBy(1);
                    setProgress(100 * progressHorizontal.getProgress());
                }
            });
            button = (Button) findViewById(R.id.decrease);
            button.setOnClickListener(new Button.OnClickListener() {
                public void onClick(View v) {
                    progressHorizontal.incrementProgressBy(-1);
                    setProgress(100 * progressHorizontal.getProgress());
                }
            });
            button = (Button) findViewById(R.id.increase_secondary);
            button.setOnClickListener(new Button.OnClickListener() {
                public void onClick(View v) {
                    progressHorizontal.incrementSecondaryProgressBy(1);
                    setSecondaryProgress(100 * progressHorizontal.getSecondaryProgress());
                }
            });
            button = (Button) findViewById(R.id.decrease_secondary);
            button.setOnClickListener(new Button.OnClickListener() {
                public void onClick(View v) {
                    progressHorizontal.incrementSecondaryProgressBy(-1);
                    setSecondaryProgress(100 * progressHorizontal.getSecondaryProgress());
                }
            });
 }
}
```

运行程序,效果如图 3-20 所示。



图 3-20　进度条实例

3.5.2 Android 滚动条

滚动条用 ScrollView 表示，用于为其他组件添加滚动条。在默认情况下，当窗体中的内容比较多而一屏幕显示不下时，超出的部分将不能被用户所看到。因为 Android 的布局管理器本身没有提供滚动屏幕的功能。如果要让其滚动，就需要使用滚动条(ScrollView)，这样用户就可以通过滚动屏幕来查看完整的内容。

滚动条是 android. widget. FrameLayout(帧布局管理器)的子类。因此，在滚动条中，可以添加任何想要放入其中的组件。但是，一个滚动条中只能放置一个组件。如果想要放置多个组件，可以先放置一个布局管理器，再将要放置的其他组件放置到该布局管理器中。在滚动条中，使用比较多的是线性布局管理器。

说明：滚动条 ScrollView 只支持垂直滚动。如果想要实现水平滚动条，可以使用水平滚动条(HorizontalScrollView)。

在 Andorid 中，可以使用两种方法向屏幕中添加滚动条，一种是通过在 XML 布局文件中使用<ScrollView>标记添加，另一种是在 Java 文件中通过 new 关键字创建。

1）在 XML 布局文件中添加

在 XML 布局文件中添加滚动视图比较简单，只需要在添加滚动条的组件外面使用下面的布局代码添加即可。

```
<ScrollView
    android:id = "@ + id/scrollView1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_below = "@ + id/textView1"
    android:layout_centerHorizontal = "true"
    android:layout_marginTop = "152dp" >
```

2）通过 new 关键字创建

在 Java 代码中，new 关键字创建滚动条比较简单，只需要经过以下步骤即可实现。

(1) 使用构造方法 ScrollView(Context context)创建一个滚动条。

(2) 创建或者获取需要添加滚动条的组件，并应用 addView()方法将其添加到滚动条中。

(3) 将滚动视图添加到整个布局管理器中，用于显示该滚动条。

下面通过一个实例来实现使用 XML 方法实现滚动条。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 ScrollView_test1。

(2) 打开 res\layout 目录下的 main. xml 布局文件，在文件中定义一个 ScrollView 组件、10 个 ImageView。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<ScrollView
xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:scrollbars = "vertical">
    <LinearLayout android:orientation = "vertical"
        android:layout_width = "fill_parent"
```

```
            android:layout_height = "fill_parent"
            android:background = "@drawable/bj1">
            < ImageView android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:src = "@drawable/face"
                android:layout_gravity = "center_horizontal"/>
            < ImageView android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:src = "@drawable/face"
                android:layout_gravity = "center_horizontal"/>
            < ImageView android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:src = "@drawable/face"
                android:layout_gravity = "center_horizontal"/>
            < ImageView android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:src = "@drawable/face"
                android:layout_gravity = "center_horizontal"/>
            < ImageView android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:src = "@drawable/face"
                android:layout_gravity = "center_horizontal"/>
            < ImageView android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:src = "@drawable/face"
                android:layout_gravity = "center_horizontal"/>
            < ImageView android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:src = "@drawable/face"
                android:layout_gravity = "center_horizontal"/>
            < ImageView android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:src = "@drawable/face"
                android:layout_gravity = "center_horizontal"/>
    < ImageView android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:src = "@drawable/face"
                android:layout_gravity = "center_horizontal"/>
        </LinearLayout >
    </ScrollView >
```

运行程序,效果如图 3-21 所示,拖动鼠标就可浏览图片。

前面例子中通过了 XML 实现滚动条,下面另外通过一个实例来使用 new 关键字实例滚动条。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 ScrollView_test2。

(2) 打开 res\layout 目录下的 main. xml 布局文件,在文件中声明一个 ScrollView 控件、一个 Button 控件以及一个 TextView 控件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< ScrollView
```



图 3-21 滚动视图效果

```
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@ + id/ScrollView"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:scrollbars = "vertical"
    android:background = "#aabbcc">
    < LinearLayout
        android:id = "@ + id/LinearLayout"
        android:orientation = "vertical"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content">
        < TextView android:id = "@ + id/TestView"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:text = "文本框 0" />
        < Button
            android:id = "@ + id/Button"
            android:text = "按钮 0"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"/>
    </LinearLayout >
</ScrollView >
```

(3) 打开 src\fs.scrollview_test2 包下的 MainActivity.java 文件，在文件中实现当单击按钮一次时，即自加一个文本框及一个按钮控件。代码为：

```
package fs.scrollview_tes2;
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
```

```java
import android.view.KeyEvent;
import android.view.View;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.ScrollView;
import android.widget.TextView;
public class MainActivity extends Activity {
    /** 第一次调用 activity 活动 */
    private LinearLayout mLayout;
    private ScrollView sView;
    private final Handler mHandler = new Handler();
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //创建一个线性布局
        mLayout = (LinearLayout) this.findViewById(R.id.LinearLayout);
        //创建一个 ScrollView 对象
        sView = (ScrollView) this.findViewById(R.id.ScrollView);
        Button mBtn = (Button) this.findViewById(R.id.Button);
        mBtn.setOnClickListener(mClickListener);       //添加单击事件监听
    }
    public boolean onKeyDown(int keyCode, KeyEvent event){
        Button b = (Button) this.getCurrentFocus();
        int count = mLayout.getChildCount();
        Button bm = (Button) mLayout.getChildAt(count - 1);
        if(keyCode == KeyEvent.KEYCODE_DPAD_UP && b.getId() == R.id.Button){
            bm.requestFocus();
            return true;
        }else if(keyCode == KeyEvent.KEYCODE_DPAD_DOWN && b.getId() == bm.getId()){
            this.findViewById(R.id.Button).requestFocus();
            return true;
        }
        return false;
    }
    //Button 事件监听,当单击第一个按钮时增加一个 Button 和一个 TextView
    private Button.OnClickListener mClickListener = new Button.OnClickListener() {
        private int index = 1;
        @Override
        public void onClick(View v) {
            TextView tView = new TextView(MainActivity.this);  //定义一个 TextView
            tView.setText("文本框" + index);                    //设置 TextView 的文本信息
                                                                //设置线性布局的属性
            LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(
                    LinearLayout.LayoutParams.FILL_PARENT,
                    LinearLayout.LayoutParams.WRAP_CONTENT);
            mLayout.addView(tView, params);                     //添加一个 TextView 控件
            Button button = new Button(MainActivity.this);      //定义一个 Button
            button.setText("按钮" + index);                     //设置 Button 的文本信息
            button.setId(index++);
            mLayout.addView(button, params);                    //添加一个 Button 控件
```

```
            mHandler.post(mScrollToButton);                    //传递一个消息进行滚动
        }
    };
    private Runnable mScrollToButton = new Runnable() {
        @Override
        public void run() {
            int off = mLayout.getMeasuredHeight() - sView.getHeight();
            if (off > 0) {
                sView.scrollTo(0,off);                         //改变滚动条的位置
            }
        }
    };
}
```

运行程序,效果如图 3-22 所示。



图 3-22　new 关键字实现滚动条效果

3.5.3　Android 拖动条

拖动条(SeekBar)与进度条类似,所不同的是,拖动条允许用户拖动滑块来改变值,通常用于实现对某种数值的调节。例如,调节图片的透明度或音量等。

在 Android 中,如果想在屏幕中添加拖动条,可以在 XML 布局文件中通过<SeekBar>标记添加,其语法格式为:

```
<SeekBar
    android:id="@+id/seekBar1"
    android:layout_width="match_parent"
```

```
    android:layout_height = "wrap_content"
    android:layout_alignParentLeft = "true"
    android:layout_below = "@ + id/textView1"
    android:layout_marginTop = "116dp" />
```

SeekBar 控件允许用户改变拖动滑块的外观，这可以使用 android:thumb 属性实现，该属性的值为一个 Drawable 对象，该 Drawable 对象将作为自定义滑块。

由于拖动条可以被用户监控，所以需要为其添加 OnSeekBarChangeListener 监听器。为拖动条添加监听器的基本代码为：

```
seekbar.setOnSeekBarChangeListener(new OnSeekBarChangeListener(){
@Override
public void onStopTrackingTouch(SeekBar seekBar){
  //要执行的代码
  }
  @Override
  public void onStartTrackingTouch(SeekBar seekBar){
  //要执行的代码
  }
  @Override
  public void onProgressChanged(SeekBar seekBar,int progress,boolean fromUser){
  //其他要执行的代码
  }
  });
```

说明：在以上代码中，onProgressChanged()方法中的参数 progress 表示当前进度，也就是拖动条的值。

下面通过一个实例来演示拖动条的用法。其具体实现步骤为：

（1）在 Eclipse 中创建一个 Android 应用项目，命名为 SeekBar_test。

（2）打开 res\layout 目录下的 main.xml 布局文件，在文件中声明一个 SeekBar 控件及两个 TextView 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
    < SeekBar
        android:id = "@ + id/SeekBar1"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:max = "100"
        android:progress = "50"
        android:secondaryProgress = "100"/>
    < TextView
        android:id = "@ + id/TextView1"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
```

```
            android:text = ""/>
        <TextView
            android:id = "@ + id/TextView2"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:text = "" />
</LinearLayout>
```

(3) 打开 src\fs. seekbar_test 包下的 MainActivity. java 文件,在文件中实现拖动,并把相关值显示在对应的 TextView 控件中。代码为:

```
package fs.seekbar_test;
import android.app.Activity;
import android.os.Bundle;
import android.widget.SeekBar;
import android.widget.TextView;
public class MainActivity extends Activity implements SeekBar.OnSeekBarChangeListener{
    /** 第 1 次调用 activity 活动 */
    private SeekBar seekBar;
    private TextView textView1,textView2;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        seekBar = (SeekBar) this.findViewById(R.id.SeekBar1);
        textView1 = (TextView) this.findViewById(R.id.TextView1);
        textView2 = (TextView) this.findViewById(R.id.TextView2);
        seekBar.setOnSeekBarChangeListener(this);              //添加事件监听
    }
    //拖动中
    @Override
    public void onProgressChanged(SeekBar seekBar,int progress,
            boolean fromUser) {
        this.textView1.setText("当前值:" + progress);
    }
    //开始拖动
    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {
        this.textView2.setText("拖动中……");
    }
    //结束拖动
    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
        this.textView2.setText("拖动完毕");
    }
}
```

运行程序,效果如图 3-23 所示。



图 3-23　拖动效果

3.5.4　Android 星级评分条

星级评分条与拖动条类似,都允许用户通过拖动来改变进度,所不同的是,星级评分条通过星星表示进度。通常使用星级评分条表示对某一事物的支持度或对某种服务的满意程度等。例如,淘宝网中对卖家的好评度,就是通过星级评分条实现的。

在 Android 中,如果想在屏幕中添加星级评分条,可以在 XML 布局文件中通过<RatingBar>标记实现,其在 XML 中用法为:

```
<RatingBar
    android:id = "@ + id/ratingBar1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_alignParentLeft = "true"
    android:layout_below = "@ + id/textView1"
    android:layout_marginTop = "74dp" />
```

RatingBar 控件支持的 XML 属性如表 3-6 所列。

表 3-6　RatingBar 支持的常见 XML 属性

XML 属性	描　　述
android:isIndicator	设置该星级评分条是否允许用户改变(true 为不允许修改)
android:numStars	设置该星级评分条总共有多少个星级
android:rating	设置该星级评分条默认的星级
android:stepSize	设置每次最少需要改变多少个星级

除此之外，星级评分条还提供了 3 个比较常用的方法。

- getRating()方法：用于获取等级，表示被选中了几颗星。
- getStepSize()方法：用于获取等级步值，表示每次选值的大小。
- getProgress()方法：用于获取进度，获取到的进度值等于 getRating()方法的返回值乘以 getStepSize()方法的返回值。

下面通过一个实例来演示 RatingBar 的用法。其具体步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 RatingBar_test。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中声明一个 TextView 控件及一个 RatingBar 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
    <TextView
        android:text = "请选择"
        android:textSize = "35dip"
        android:id = "@ + id/TextView1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content">
    </TextView>
    <RatingBar
        android:id = "@ + id/RatingBar1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content">
    </RatingBar>
</LinearLayout>
```

(3) 打开 src\fs.ratingbar_test 包下的 MainActivity.java 文件，在文件中实现评分功能，并把打分结果显示在 TextView 控件中。代码为：

```
package fs.ratingbar_test;
import android.app.Activity;
import android.os.Bundle;
import android.widget.RatingBar;
import android.widget.TextView;
public class MainActivity extends Activity
{
    RatingBar rb;
    TextView tv;
    @Override
public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        rb = (RatingBar)this.findViewById(R.id.RatingBar1);
        tv = (TextView)this.findViewById(R.id.TextView1);
```

```
            rb.setOnRatingBarChangeListener
            (
                    new RatingBar.OnRatingBarChangeListener()
                    {
                        public void onRatingChanged(RatingBar ratingBar,float rating,boolean fromUser)
                        {
                            tv.setText("您的打分为:" + (float)rb.getRating() + "分");
                        }
                    }
            );
        }
    }
```

运行程序,效果如图 3-24 所示。



图 3-24　星级评分条效果

3.6　Android 时钟控件

本节将对 Android 中的时钟控件进行介绍,时钟控件是 Android 用户界面中比较简单的控件,时钟控件包括 AnalogClock 控件和 DigitalClock 控件。

时钟 UI 组件是两个非常简单的组件,DigitalClock 本身就继承了 TextView,也就是说其本身就是文本框,只是它里面显示的内容是当前时间;AnalogClock 则继承了 View 组件,其重写了 View 的 OnDraw 方法,会在 View 上显示模拟时钟。

DigitalClock 和 AnalogClock 都会显示当前时间。不同的是,DigitalClock 显示数字时钟,可以显示当前的秒数;而 AnalogClock 则显示模拟时钟,不会显示当前秒数。

AnalogClock 和 DigitalClock 的继承关系如图 3-25 所示。

```
java.lang.Object
  └──── android.view.View
         └──── android.widget.AnalogClock
                └────android.widget.TextView
                       └──── android.widget.DigitalClock
```

图 3-25 AnalogClock 和 DigtialClock 类的继承关系

它们在具体实现上，使用到如下 3 个对象。

(1) android. os. Handler：通过产生的 Thread 对象在进程内同步调用方法 System. currentTimeMillis()，这样可以获得系统时间。

(2) java. lang. Thread：为联系 Activity 与 Thread 的桥梁。

(3) android. os. Message：使用 Message 对象通知 Handler 对象，在收到 Message 对象后将时间变量的值显示在 TextView 中，这样就实现了数字时钟功能。

下面通过一个实例来演示时钟控件的用法。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Clock_test。

(2) 打开 res\layout 目录下的 main. xml 布局文件，布局一个文本框及一个模拟时钟控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout
  android:id = "@ + id/widget27"
  android:layout_width = "fill_parent"
  android:layout_height = "fill_parent"
  xmlns:android = "http://schemas.android.com/apk/res/android"
  android:orientation = "vertical"
  android:background = "@drawable/bj1">
< AnalogClock
  android:id = "@ + id/myAnalogClock"
  android:layout_width = "wrap_content"
  android:layout_height = "wrap_content"
  android:layout_gravity = "center_horizontal">
</AnalogClock >
< TextView
  android:id = "@ + id/myTextView"
  android:layout_width = "wrap_content"
  android:layout_height = "wrap_content"
  android:text = "TextView"
  android:textSize = "20sp"
  android:textColor = "@drawable/white"
  android:layout_gravity = "center_horizontal">
</TextView >
</LinearLayout >
```

(3) 在 res\value 目录中，创建一个名为 color. xml 的文件，用于实现文本框的颜色。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < drawable name = "white"> # FF1F0F </drawable >
</resources >
```

(4) 打开 src\fs. clock_test 目录下的 MainActivity. java 文件，实现时钟组件。代码为：

```
import android.app.Activity;
import android.os.Bundle;
/* 这里需要使用 Handler 类与 Message 类来处理运行线程 */
import android.os.Handler;
import android.os.Message;
import android.widget.AnalogClock;
import android.widget.TextView;
/* 需要使用 Java 的 Calendar 与 Thread 类来获得系统时间 */
import java.util.Calendar;
import java.lang.Thread;
public class MainActivity extends Activity
{
  /* 声明一常数作为判别信息用 */
  protected static final int GUINOTIFIER = 0x1234;
  /* 声明两个 widget 对象变量 */
  private TextView mTextView;
  public AnalogClock mAnalogClock;
  /* 声明与时间相关的变量 */
  public Calendar mCalendar;
  public int mMinutes;
  public int mHour;
  /* 声明关键 Handler 与 Thread 变量 */
  public Handler mHandler;
  private Thread mClockThread;
  /** 第 1 次调用活动 */
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    /* 通过 findViewById 取得两个 widget 对象 */
    mTextView = (TextView)findViewById(R.id.myTextView);
    mAnalogClock = (AnalogClock)findViewById(R.id.myAnalogClock);
    /* 通过 Handler 来接收运行线程所传递的信息并更新 TextView */
    mHandler = new Handler()
    {
      public void handleMessage(Message msg)
      {
        /* 这里是处理信息的方法 */
        switch (msg.what)
        {
          case MainActivity.GUINOTIFIER:
          /* 在这处理要 TextView 对象 Show 时间的事件 */
            mTextView.setText(mHour + " : " + mMinutes);
            break;
        }
        super.handleMessage(msg);
      }
    };
    /* 通过运行线程来持续取得系统时间 */
      mClockThread = new LooperThread();
      mClockThread.start();
  }
```

```
/* 改写一个 Thread Class 用来持续获得系统时间 */
class LooperThread extends Thread
{
  public void run()
  {
    super.run();
    try
    {
      do
      {
        /* 获得系统时间 */
        long time = System.currentTimeMillis();
        /* 通过 Calendar 对象来获得小时与分钟 */
        final Calendar mCalendar = Calendar.getInstance();
        mCalendar.setTimeInMillis(time);
        mHour = mCalendar.get(Calendar.HOUR);
        mMinutes = mCalendar.get(Calendar.MINUTE);
        /* 让运行线程休息一秒 */
        Thread.sleep(1000);
          /* 重要关键程序:获得时间后发出信息给 Handler */
          Message m = new Message();
          m.what = MainActivity.GUINOTIFIER;
          MainActivity.this.mHandler.sendMessage(m);
      }while(MainActivity.LooperThread.interrupted() == false);
      /* 当系统发出中断信息时停止本循环 */
    }
catch(Exception e)
    {
      e.printStackTrace();
    }
  }
 }
}
```

运行程序,效果如图 3-26 所示。



图 3-26　时钟控件

3.7 Android 日期时间控件

本节将介绍日期 DatePicker 与时间 TimePicker 选择控件。

3.7.1 Android 日期选择控件

DatePicker(日期控件)类继承自 FrameLayout 类,日期选择控件主要的功能向用户提供包含了年月日的日期数据并允许用户对其进行选择。如果要捕获用户修改日期选择控件中数据的事件,需要为 DatePicker 添加 onDateChangedListener 监听器。DatePicker 类的主要成员方法如表 3-7 所列。

表 3-7 DatePicker 类主要的成员方法及说明

名　　称	说　　明
getDayOfMonth()	获取日期天数
getMonth()	获取日期月份
getYear()	获取日期年份
init(int year, int monthOfYear, int dayOfMonth, DatePicker. OnDateChangedListener onDateChangedListener)	初始化 DatePicker 控件的属性,参数 onDateChangedListener 为监听器对象,负责监听日期数据的变化
setEnabled(boolean enabled)	根据传入的参数设置日期选择控件是否可用
updateDate (int year, int monthOfYear, int dayOfMonth)	根据传入的参数更新日期选择控件的各个属性值

下面通过一个实例来演示日期控件的使用。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 DatePicker_test。

(2) 打开 res\layout 目录下的 main. xml 文件,在文件中声明一个 TimePicker 控件及一个 Button 控件。代码为:

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc">
<TimePicker
        android:id = "@ + id/timePic1"
        android:layout_height = "wrap_content"
        android:layout_width = "match_parent"/>
    <Button
        android:id = "@ + id/buttone1"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:layout_below = "@id/timePic1"
```

```
        android:text = "获取时间"/>
</RelativeLayout>
```


(3) 打开 src\fs. timepicker_test 包下的 MainActivity. java 文件，在文件中实现时间选择，当单击按钮时，即获取所选择的时间。代码为：

```
package fs.datepicker_test;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TimePicker;
import android.widget.TimePicker.OnTimeChangedListener;
public class MainActivity extends Activity {
    private TimePicker timePick1;
    private Button buttone1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        timePick1 = (TimePicker)findViewById(R.id.timePic1);
        buttone1 = (Button)findViewById(R.id.buttone1);
        OnChangeListener  buc = new OnChangeListener();
        buttone1.setOnClickListener(buc);
        //是否使用 24 小时制
        timePick1.setIs24HourView(true);
        TimeListener times = new TimeListener();
        timePick1.setOnTimeChangedListener(times);
    }
    class OnChangeListener implements OnClickListener{
        @Override
        public void onClick(View v) {
            //TODO 自动存根法
            int h = timePick1.getCurrentHour();
            int m = timePick1.getCurrentMinute();
            System.out.println("h:" + h + "  m:" + m);
        }
    }
    class TimeListener implements OnTimeChangedListener{
        /**
          * view 当前选中 TimePicker 控件
          *  hourOfDay 当前控件选中 TimePicker 的小时
          * minute 当前选中控件 TimePicker 的分钟
          */
        @Override
        public void onTimeChanged(TimePicker view, int hourOfDay, int minute) {
            //TODO 自动存根法
            System.out.println("h:" + hourOfDay + " m:" + minute);
        }
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
```

```
            getMenuInflater().inflate(R.menu.main,menu);
            return true;
        }
    }
```

运行程序,效果如图 3-27 所示。



图 3-27　时间选择控件效果

3.7.2　Android 时间选择控件

TimePicker(时间选择控件)同样继承自 FrameLayout 类,时间选择控件向用户显示一天中的时间(可以为 24 小时制,也可以为 AM/PM 制),并允许用户进行选择。如果要捕获用户修改时间数据的事件,即需要为 TimePicker 添加 OnTimeChangedListener 监听器。TimePicker 类的主要成员方法如表 3-8 所示。

表 3-8　TimePicker 类主要的成员方法及说明

名　　称	说　　明
getCurrentHour()	获取时间选择控件的当前小时,返回 Integer 对象
getCurrentMinute()	获取时间选择控件的当前分钟,返回 Integer 对象
is24HourView()	判断时间选择控件是否为 24 小时制
setCurrentHour(Integer currentHour)	设置时间选择控件的当前小时,传入 Integer 对象
setCurrentMinute(Integer currentMinute)	设置时间选择控件的当前分钟,传入 Integer 对象
setEnabled(boolean enabled)	根据传入的参数设置时间选择控件是否可用
setIs24HourView(boolean is24HourView)	设置时间是否为 24 小时制
setOnTimeChangedListener(TimePicker.OnTimeChangedListener onTimeChangedListener)	为时间选择控件添加 OnTimeChangedListener 监听器

下面通过一个实例来演示时间选择控件的用法。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 TimePicker_test。

(2) 打开 res\layout 目录下的 main. xml 布局文件，在文件中声明一个 DatePicker 控件及一个 Button 控件。代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc">
    <DatePicker
        android:id = "@ + id/datePick1"
        android:layout_height = "wrap_content"
        android:layout_width = "match_parent" />
    <Button
        android:id = "@ + id/button1"
        android:layout_below = "@id/datePick1"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:text = "获取日期"/>
</RelativeLayout>
```

(3) 打开 src\fs. timepicker_test 包下的 MainActivity. java 文件，在文件中实现时间的选择，当单击按钮控件时获取对应的时间值。代码为：

```
package fs.timepicker_test;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.DatePicker;
public class MainActivity extends Activity {
    private DatePicker datePicker1;
    private Button button1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        datePicker1 = (DatePicker)findViewById(R.id.datePick1);
        //设置默认的时间,例如 2055 年 9 月 9 日
        datePicker1.updateDate(2012,8,9);
        button1 = (Button)findViewById(R.id.button1);
        OnClicLisers   cl = new OnClicLisers();
        button1.setOnClickListener(cl);
    }
    class OnClicLisers implements OnClickListener{
```

```
            @Override
            public void onClick(View v) {
                //TODO 自动存根法
                int y = datePicker1.getYear();
                int m = datePicker1.getMonth() + 1;
                int d = datePicker1.getDayOfMonth();
                System.out.println("y:" + y + " m:" + m + " d:" + d);
            }
        }
        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            getMenuInflater().inflate(R.menu.main,menu);
            return true;
        }
    }
```

运行程序,效果如图 3-28 所示。



图 3-28　时间选择控件实例

3.8　Android 计时器

计时器组件可实现显示从某个起始时间开始,一共过去了多长时间的文本,使用 Chronometer 表示。由于该组件继承自 TextView,所以它将以文本的形式显示内容。该组件比较简单,通常只需要使用以下 5 个方法。

- setBase():用于设置计时器的起始时间。
- setFormat():用于设置显示时间的格式。

- start()：用于指定开始计时。
- stop()：用于指定停止计时。
- setOnChronometerTickListener()：用于为计时器绑定事件监听器，当计时器改变时触发该监听器。

说明：默认情况下，计时器返回的值为 MM:SS 的格式，例如，8 分零 12 秒将显示为 08:12 的形式。在使用 setFormat()方法设置显示时间的格式时，可以使用%s 表示计时信息，例如，要设置显示时间的格式为"已用时间：MM:SS"，可以将 setFormat()的参数设置为"已用时间：%s"。

下面通过一个实例来演示计时器的使用。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Chronometer_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明一个 Chronometer 控件及 3 个 Button 控件。代码为：

```
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:background = "#aabbcc">
<Chronometer
    android:id = "@ + id/chronometer"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"/>
    <LinearLayout
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:background = "#aabbcc">
        <Button
            android:onClick = "onStart"
            android:text = "计时开始"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"/>
        <Button
            android:onClick = "onStop"
            android:text = "计时停止"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"/>
        <Button
            android:onClick = "onReset"
            android:text = "重置"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"/>
    </LinearLayout>
</LinearLayout>
```

(3) 打开 src\fs.chronometer_test 包下的 MainActivity.java 文件，在文件中实现计时开始、计时停止及计时重置等功能。代码为：

```
package fs.chronometer_test;
import android.app.Activity;
import android.os.Bundle;
```

```
import android.os.SystemClock;
import android.view.View;
import android.widget.Chronometer;
public class MainActivity extends Activity
{
    private Chronometer chronometer = null;

    /** 第 1 次调用 activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        chronometer = (Chronometer) findViewById(R.id.chronometer);
        chronometer.setFormat("计时: %s");
    }
    public void onStart(View view)
    {
          chronometer.start();
    }
    public void onStop(View view)
    {
          chronometer.stop();
    }
    public void onReset(View view)
    {
          chronometer.setBase(SystemClock.elapsedRealtime());
    }
}
```

运行程序,效果如图 3-29 所示。



图 3-29　计时器实例

3.9 Android 控件综合实例

前面内容已对 Android 的控件做了基本介绍，并都给出相应的案例来进行说明，本节将综合利用基本控件实现一个登录界面，对于绝大部分的应用来说都是有必要的。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Comprehensive_test。

(2) 打开 res\layout 目录下的 main.xml 文件。代码为：

```
<! -- 声明一个相对布局 -- >
< RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    tools:context = ".MainActivity"
    android:background = "#aabbcc">
    <! -- 声明一个图片控件 -- >
    < ImageView
        android:id = "@ + id/loginbutton"
        android:layout_width = "120px"
        android:layout_height = "120px"
        android:layout_centerHorizontal = "true"
        android:src = "@drawable/b3" />
    <! -- 声明一个线性布局 -- >
    < LinearLayout
        android:id  = "@ + id/input"
        android:layout_width  = "fill_parent"
        android:layout_height  = "wrap_content"
        android:layout_below  = "@id/loginbutton"
        android:layout_marginLeft  = "28.0dip"
        android:layout_marginRight  = "28.0dip"
        android:orientation  = "vertical" >
      <! -- 声明一个线性布局 -- >
        < LinearLayout
            android:layout_width  = "fill_parent"
            android:layout_height  = "44.0dip"
            android:gravity  = "center_vertical"
            android:orientation  = "horizontal">
          <! -- 声明一个 EditText 控件 -- >
            < EditText
                android:id  = "@ + id/searchEditText"
                android:layout_width  = "0dp"
                android:layout_height  = "fill_parent"
                android:layout_weight  = "1"
                android:background  = "@null"
                android:ems  = "10"
                android:imeOptions  = "actionDone"
                android:hint = "用户名"
                android:singleLine  = "true"
                android:textSize  = "16sp"/>
        <! -- 声明一个 Button 控件 -- >
```

```
            <Button
                android:id = "@ + id/button_clear"
                android:layout_width = "20dip"
                android:layout_height = "20dip"
                android:layout_marginRight = "8dip"
                android:visibility = "visible"/>
        </LinearLayout>
    <! -- 声明一个 View 控件 -->
        <View
            android:layout_width = "fill_parent"
            android:layout_height = "1.0px"
            android:layout_marginLeft = "1.0px"
            android:layout_marginRight = "1.0px"
            android:background = "#ffc0c3c4" />
      <! -- 声明一个 EditText 控件 -->
        <EditText
            android:id = "@ + id/password"
            android:layout_width = "fill_parent"
            android:layout_height = "44.0dip"
            android:background = "#00ffffff"
            android:gravity = "center_vertical"
            android:inputType = "textPassword"
            android:maxLength = "16"
            android:maxLines = "1"
            android:textColor = "#ff1d1d1d"
            android:textColorHint = "#ff666666"
            android:hint = "密码"
            android:textSize = "16.0sp" />
    </LinearLayout>
<! -- 声明一个 Button 控件 -->
    <Button
        android:id = "@ + id/buton1"
        android:layout_width = "270dp"
        android:paddingTop = "5.0dip"
        android:layout_height = "50dp"
        android:layout_marginLeft = "28.0dip"
        android:layout_marginRight = "28.0dip"
        android:layout_marginTop = "12.0dip"
        android:layout_below = "@ + id/input"
        android:gravity = "center"
        android:textSize = "20dp"
        android:text = "登录" />
<! -- 声明一个相对布局 -->
    <RelativeLayout
        android:id = "@ + id/relative"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/input"
        android:layout_alignRight = "@ + id/input"
        android:layout_below = "@id/buton1" >
<! -- 声明一个 CheckBox 控件 -->
      <CheckBox
            android:id = "@ + id/auto_save_password"
            android:layout_width = "wrap_content"
```

```
                android:layout_height = "wrap_content"
                android:layout_alignParentLeft = "true"
                android:checked = "true"
                android:drawablePadding = "4.0dip"
                android:text = "记住密码"
                android:textSize = "12.0sp" />
          <! -- 声明一个 Button 控件 -->
            < Button
                android:id = "@ + id/regist"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:layout_alignParentRight = "true"
                android:clickable = "true"
                android:gravity = "left|center"
                android:paddingLeft = "8.0dip"
                android:paddingRight = "18.0dip"
                android:text = "注册新账号"
                android:textSize = "12.0sp" />
        </RelativeLayout >
    <! -- 声明一个线性布局 -->
        < LinearLayout
            android:id = "@ + id/more_bottom"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:layout_alignParentBottom = "true"
            android:orientation = "vertical" >
    <! -- 声明一个相对布局 -->
        < RelativeLayout
            android:id = "@ + id/input2"
            android:layout_width = "fill_parent"
            android:layout_height = "40dp"
            android:background = "#868686"
            android:orientation = "vertical" >
    <! -- 声明一个 TextView 控件 -->
          < TextView
                android:id = "@ + id/more_text"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:layout_centerInParent = "true"
                android:background = "@null"
                android:gravity = "center"
                android:maxLines = "1"
                android:text = "更多登录选项"
                android:textColor = "#ffccee"
                android:textSize = "14.0sp" />
        </RelativeLayout >
          <! -- 声明一个线性布局 -->
          < LinearLayout
                android:id = "@ + id/morehidebottom"
                android:layout_width = "fill_parent"
                android:layout_height = "wrap_content"
                android:orientation = "vertical"
            android:background = "#868686"
                android:visibility = "gone" >
```

```
    <!-- 声明一个 View 控件 -->
      <View
                android:layout_width = "fill_parent"
                android:layout_height = "1.0px"
                android:background = "#ff005484" />
  <!-- 声明一个 View 控件 -->
    <View
                android:layout_width = "fill_parent"
                android:layout_height = "1.0px"
                android:background = "#ff0883cb" />
            <!-- 声明一个线性布局 -->
          <LinearLayout
                android:layout_width = "fill_parent"
                android:layout_height = "wrap_content"
                android:layout_marginLeft = "30.0dip"
                android:layout_marginRight = "30.0dip"
                android:layout_marginTop = "12.0dip"
                android:orientation = "horizontal" >
                <!-- 声明一个 CheckBox 控件 -->
                <CheckBox
                    android:id = "@+id/hide_login"
                    android:layout_width = "1.0px"
                    android:layout_height = "wrap_content"
                    android:layout_weight = "2.0"
                    android:checked = "false"
                    android:drawablePadding = "4.0dip"
                    android:text = "隐身登录"
                    android:textColor = "#ffccee"
                    android:textSize = "12.0sp" />
                  <!-- 声明一个 CheckBox 控件 -->
                <CheckBox
                    android:id = "@+id/silence_login"
                    android:layout_width = "1.0px"
                    android:layout_height = "wrap_content"
                    android:layout_weight = "1.0"
                    android:checked = "false"
                    android:drawablePadding = "4.0dip"
                    android:text = "静音登录"
                    android:textColor = "#ffccee"
                    android:textSize = "12.0sp" />
          </LinearLayout>
      <!-- 声明一个线性布局 -->
          <LinearLayout
                android:layout_width = "fill_parent"
                android:layout_height = "wrap_content"
                android:layout_marginBottom = "18.0dip"
                android:layout_marginLeft = "30.0dip"
                android:layout_marginRight = "30.0dip"
                android:layout_marginTop = "18.0dip"
                android:orientation = "horizontal"
                android:background = "#868686">
                <!-- 声明一个 CheckBox 控件 -->
                <CheckBox
                    android:id = "@+id/accept_accounts"
```

```
                android:layout_width = "1.0px"
                android:layout_height = "wrap_content"
                android:layout_weight = "2.0"
                android:checked = "true"
                android:drawablePadding = "4.0dip"
                android:singleLine = "true"
                android:text = "允许手机/计算机同时在线"
                android:textColor = "#ffccee"
                android:textSize = "12.0sp" />
            <!-- 声明一个 CheckBox 控件 -->
            <CheckBox
                android:id = "@+id/accept_troopmsg"
                android:layout_width = "1.0px"
                android:layout_height = "wrap_content"
                android:layout_weight = "1.0"
                android:checked = "true"
                android:drawablePadding = "4.0dip"
                android:text = "接收系统消息"
                android:textColor = "#ffccee"
                android:textSize = "12.0sp" />
        </LinearLayout>
      </LinearLayout>
    </LinearLayout>
</RelativeLayout>
```

(3) 打开 src\fs.comprehensive_test 目录下的 MainActivity.java 文件，在文件中实现界面的登录。代码为：

```
package fs.comprehensive_test;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageView;
public class MainActivity extends Activity implements OnClickListener{
    private Button login_Button;
    private View moreHideBottomView,input2;
    private boolean mShowBottom = false;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        initView();
    }
    private void initView() {
        login_Button = (Button) findViewById(R.id.buton1);
        login_Button.setOnClickListener(this);
        moreHideBottomView = findViewById(R.id.morehidebottom);
        input2 = findViewById(R.id.input2);
        input2.setOnClickListener(this);
    }
    public void showBottom(boolean bShow){
```

```
            if(bShow){
                moreHideBottomView.setVisibility(View.GONE);
                mShowBottom = true;
            }else{
                moreHideBottomView.setVisibility(View.VISIBLE);
                mShowBottom = false;
            }
        }
    public void onClick(View v) {
            switch(v.getId())
            {
            case R.id.input2:
                showBottom(!mShowBottom);
                break;
                default:
                    break;
            }
        }
        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            getMenuInflater().inflate(R.menu.main,menu);
            return true;
        }
}
```

运行程序，效果如图 3-30 所示。





图 3-30　实现登录界面

第 4 章　Android 对话框与菜单

菜单和对话框是应用程序 UI 设计不可或缺的一部分。Android 系统中提供了 3 种形式的菜单，包括选项菜单（Options Menu）、子菜单（SubMenu）和上下文菜单（Content Menu），开发者可以为应用程序添加选项菜单，也可以为界面上的一个 View 对象添加上下文菜单。Android 系统同时还提供了多种形式的对话框，可以加强用户与应用程序之间的交互。

4.1　Android 对话框

在 Android 开发中，我们经常会需要在 Android 界面上弹出一些对话框，例如询问用户或者让用户选择。这些功能称它为 Android Dialog 对话框，在使用 Android 的过程中，Android Dialog 的类型有 9 类，下面分别介绍这 9 种 Android Dialog 对话框的使用方法。

4.1.1　对话框概述

对话框是 Activity 运行时显示的小窗口，当显示对话框时，当前 Activity 失去焦点而由对话框负责所有的人机交互。一般来说，对话框用于提示消息或弹出一个与程序主进程直接相关的小程序。

对话框是作为 Activity 的一部分被创建和显示的，在程序中通过开发回调方法 onCreateDialog 来完成对话框的创建，该方法需要传入代表对话框的 id 参数。如果需要显示对话框，则调用 showDialog 方法传入对话框的 id 来显示指定的对话框。

当对话框第 1 次被显示时，Android 会调用 onCreateDialog 方法来创建对话框实例，之后将不再重复创建该实例，这点和被选菜单比较类似。同时，每次对话框在被显示之前都会调用 onPrepareDialog 方法，如果不重写该方法，那么每次显示的对话框将会是最初创建的那个。

当需要关闭对话框时，可以调用 Dialog 类的 dismiss 方法来实现，但是要注意的是以这种方式关闭的对话框并不会彻底消失，Android 会在后台保留其状态。如果需要让对话框在关闭之后彻底被清除，要调用 removeDialog 方法并传入 Dialog 的 id 值来彻底释放对话框。

提示：如果需要在调用 dismiss 方法关闭对话框时执行一些特定的工作，则可以为对话框设置 OnDismissListener 并重写其中的 onDismiss 方法来开发特定的功能。

4.1.2 AlertDialog 类对话框

Activities 提供了一种方便管理的创建、保存、回复的对话框机制，例如 onCreateDialog(int)、onPrepareDialog(int, Dialog)、showDialog(int)、dismissDialog(int)等方法，如果使用这些方法的话，Activity 将通过 getOwnerActivity()方法返回该 Activity 管理的对话框(dialog)。

- onCreateDialog(int)：当使用这个回调函数时，Android 系统会有效地设置这个 Activity 为每个对话框的所有者，从而自动管理每个对话框的状态并挂靠到 Activity 上。这样，每个对话框继承这个 Activity 的特定属性。例如，当一个对话框打开时，菜单键显示为这个 Activity 定义的选项菜单，音量键修改 Activity 使用的音频流。
- showDialog(int)：当想要显示一个对话框时，调用 showDialog(intid)方法并传递一个唯一标识这个对话框的整数。当对话框第 1 次被请求时，Android 从 Activity 中调用 onCreateDialog(intid)，应该在这里初始化这个对话框 Dialog。这个回调方法被传以和 showDialog(intid)有相同的 ID。当创建这个对话框后，在 Activity 的最后返回这个对象。
- onPrepareDialog(int, Dialog)：在对话框被显示之前，Android 还调用了可选的回调函数 onPrepareDialog(intid, Dialog)。如果想在每一次对话框被打开时改变它的任何属性，可定义这个方法。该方法在每次打开对话框时被调用，而 onCreateDialog(int)仅在对话框第 1 次打开时被调用。如果不定义 onPrepareDialog()，那么这个对话框将保持和上次打开时一样。这个方法也被传递以对话框的 ID，和在 onCreateDialog()中创建的对话框对象。
- dismissDialog(int)：当准备关闭对话框时，可以通过对这个对话框调用 dismiss()来消除它。如果需要，还可以从这个 Activity 中调用 dismissDialog(intid)方法，这实际上对这个对话框调用 dismiss()方法。如果想使用 onCreateDialog(intid)方法来管理对话框的状态(就如同在前面的章节讨论的那样)，然后每次对话框消除的时候，这个对话框对象的状态将由该 Activity 保留。如果决定不再需要这个对象或者清除该状态是重要的，那么应该调用 removeDialog(intid)。这将删除任何内部对象引用而且如果这个对话框正在显示，它将被消除。

使用 AlertDialog 可以生成的对话框概括起来有以下 4 种。

① 带“确定”、“中立”和“取消”等 N 个按钮的提示对话框，其中的按钮个数不是固定的，可以根据需要添加。例如，不需要有“中立”按钮，那么就可以生成只带有“确定”和“取消”按钮的对话框，也可以是只带有一个按钮的对话框。

② 带列表的列表对话框。

③ 带多个单选列表项和 N 个按钮的列表对话框。

④ 带多个多选列表项和 N 个按钮的列表对话框。

在使用 AlertDialog 类生成对话框时，常用的方法如表 4-1 所示。

表 4-1　AlertDialog 类的常用方法

方　　法	描　　述
setTitle(CharSequence title)	用于为对话框设置标题
setIcon(Drawable icon)	用于为对话框设置图标
setIcon(int resId)	用于设置对话框图标 Id
setMessage(CharSequence message)	用于为提示对话框设置要显示的内容
setButton()	用于为提示对话框添加按钮,可以是“取消”按钮、“中立”按钮和“确定”按钮。需要通过为其指定 int 类型的 whichButton 参数实现,其参数值可以是 DialogInterface. BUTTON_POSITIVE(“确定”按钮)、BUTTON_NEGATIVE(“取消”按钮)或 BUTTON_NEUTRAL(“中立”按钮)

通常情况下,使用 AlertDialog 类只能生成带 *N* 个按钮的提示对话框,要生成另外 3 种列表对话框,需要使用 AlertDialog. Builder 类,该类提供的常用方法如表 4-2 所示。

表 4-2　AlertDialog. Builder 类的常用方法

方　　法	描　　述
setTitle(CharSequence title)	用于为对话框设置标题
setIcon(Drawable icon)	用于为对话框设置图标
setIcon(int resId)	用于为对话框设置图标
setMessage(CharSequence message)	用于为提示对话框设置要显示的内容
setNegativeButton()	用于为对话框添加“取消”按钮
setPositiveButton()	用于为对话框添加“确定”按钮
setNeutralButton()	用于为对话框添加“中立”按钮
setItems()	用于为对话框添加列表项
setSingleChoiceItems()	用于为对话框添加单选列表项
setMultiChoiceItems()	用于为对话框添加多选列表项

下面通过一个简单实例来创建一个简单对话框。

【例 4-1】　一个简单的对话框。代码为:

```
package com.example.alertdialog_t;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.os.Bundle;
public class MainActivity extends Activity
{
    /** 第 1 次调用活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Dialog alertDialog = new AlertDialog.Builder(this).
                setTitle("对话框的标题")
```

```
                setMessage("对话框的内容")
                setIcon(R.drawable.ic_launcher)
                create();
        alertDialog.show();
    }
}
```

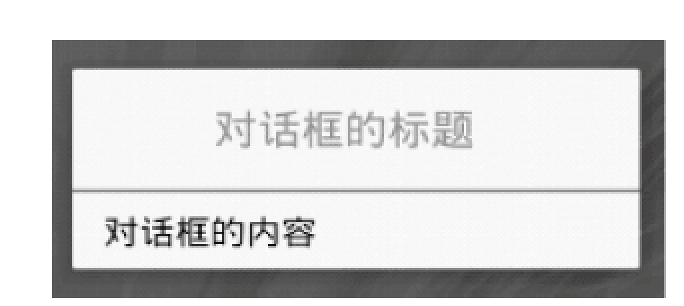


图 4-1 简单对话框

运行程序，效果如图 4-1 所示。

下面再通过一个具体的实例来说明怎样应用 AlertDialog 类生成各种提示对话框和列表对话框。

【例 4-2】 创建各种类型的对话框。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 AlertDialog_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件，将默认添加的 TextView 控件删除，然后添加 4 个控件用于各种对话框显示的按钮。代码为：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#aabbcc">
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="显示带按钮的对话框" />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="显示带列表的对话框" />
    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="显示带单选列表项的对话框" />
    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="显示带多选列表项的对话框" />
</LinearLayout>
```

(3) 打开 src\fs.alertdialog_test 包下的 MainActivity.java 文件，在文件中实现当单击对应的按钮时，即显示对应类型的对话框。代码为：

```
package fs.alertdialog_test;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.AlertDialog.Builder;
```

```
import android.content.DialogInterface;
import android.content.DialogInterface.OnClickListener;
import android.content.DialogInterface.OnMultiChoiceClickListener;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
public class MainActivity extends Activity {
    private boolean[] checkedItems;                         //记录各列表项的状态
    private String[] items;                                 //各列表项要显示的内容
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button button1 = (Button) findViewById(R.id.button1);
                                    //获取"显示带取消、中立和确定按钮的对话框"按钮
        //为"显示带取消、中立和确定按钮的对话框"按钮添加单击事件监听器
        button1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                AlertDialog alert = new AlertDialog.Builder(MainActivity.this).create();
                alert.setIcon(R.drawable.ic_launcher);          //设置对话框的图标
                alert.setTitle("系统提示：");                   //设置对话框的标题
        alert.setMessage("带取消、中立和确定按钮的对话框!");    //设置要显示的内容
                //添加"取消"按钮
                alert.setButton(DialogInterface.BUTTON_NEGATIVE,"取消",new OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        Toast.makeText(MainActivity.this,"您单击了取消按钮",
                                Toast.LENGTH_SHORT).show();
                    }
                });
                //添加"确定"按钮
                alert.setButton(DialogInterface.BUTTON_POSITIVE,"确定",new OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog,int which) {
                        Toast.makeText(MainActivity.this,"您单击了确定按钮",
                                Toast.LENGTH_SHORT).show();
                    }
                });
                alert.setButton(DialogInterface.BUTTON_NEUTRAL,"中立",new OnClickListener(){
                    @Override
                    public void onClick(DialogInterface dialog,int which) {}
                });                                             //添加"中立"按钮
                alert.show();                                   //创建对话框并显示
            }
        });
        //带列表的对话框
        Button button2 = (Button) findViewById(R.id.button2);
                                                //获取"显示带列表的对话框"按钮
        button2.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {
        final String[] items = new String[] { "游泳","羽毛球","跳高","跑步","体操" };
                Builder builder = new AlertDialog.Builder(MainActivity.this);
                builder.setIcon(R.drawable.ic_launcher);        //设置对话框的图标
                builder.setTitle("请选择你喜欢的运动类型:"); //设置对话框的标题
                                                                //添加列表项
                builder.setItems(items,new OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog,int which) {
                        Toast.makeText(MainActivity.this,
                                "您选择了" + items[which],Toast.LENGTH_SHORT).show();
                    }
                });
                builder.create().show();                        //创建对话框并显示
            }
        });
        //带多个单选列表和"确定"按钮的列表对话框
        Button button3 = (Button) findViewById(R.id.button3);
        //获取"显示带单选列表项的对话框"按钮
        button3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
        final String[] items = new String[] { "标准","无声","会议","户外",    "离线" };
                //显示带单选列表项的对话框
                Builder builder = new AlertDialog.Builder(MainActivity.this);
                builder.setIcon(R.drawable.ic_launcher);        //设置对话框的图标
                builder.setTitle("请选择要使用的情景模式:"); //设置对话框的标题
                builder.setSingleChoiceItems(items,0,new OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog,int which) {
                        Toast.makeText(MainActivity.this,
        "您选择了" + items[which],Toast.LENGTH_SHORT).show(); //显示选择结果
                    }
                });
                builder.setPositiveButton("确定",null);         //添加"确定"按钮
                builder.create().show();                        //创建对话框并显示
            }
        });
        //带多个多选列表和"确定"按钮的列表对话框
        Button button4 = (Button) findViewById(R.id.button4);
    //获取"显示带多选列表项的对话框"按钮
        button4.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
              //记录各列表项的状态
                checkedItems = new boolean[] { false,true,false,true,false };
              //各列表项要显示的内容
                items = new String[] { "桃子","石榴","香蕉","葡萄","橙子" };
                                                               //显示带单选列表项的对话框
                Builder builder = new AlertDialog.Builder(MainActivity.this);
```

```
                    builder.setIcon(R.drawable.ic_launcher);        //设置对话框的图标
                    builder.setTitle("请选择您喜爱的水果：");        //设置对话框标题
                    builder.setMultiChoiceItems(items,checkedItems,
                            new OnMultiChoiceClickListener() {
                                @Override
                                public void onClick(DialogInterface dialog,
                                        int which,boolean isChecked) {
                        checkedItems[which] = isChecked;            //改变被操作列表项的状态

                                }
                            });
                                                                    //为对话框添加"确定"按钮
                    builder.setPositiveButton("确定",new OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog,int which) {
                            String result = "";                     //用于保存选择结果
                            for(int i = 0;i < checkedItems.length;i++){
                                if(checkedItems[i]){                //当选项被选择时
                                result += items[i] + "、";          //将选项的内容添加到 result 中
                                }
                            }
                            //当 result 不为空时,通过消息提示框显示选择的结果
                            if(!"".equals(result)){
                result = result.substring(0,result.length() - 1);   //去掉最后面添加的"、"号
                                Toast.makeText(MainActivity.this,"您选择了[ " + result + " ]",
    Toast.LENGTH_LONG).show();
                            }
                        }
                    });
                    builder.create().show();                        //创建对话框并显示
                }
            });
        }
    }
```

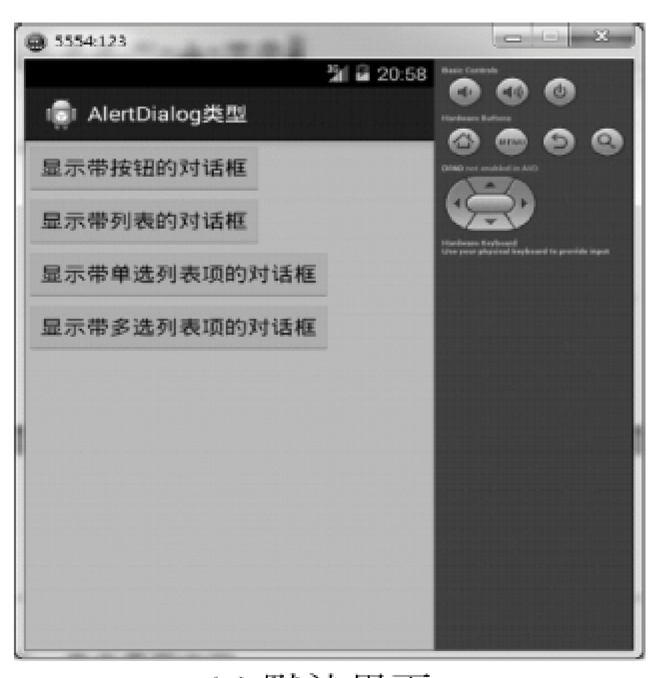
运行程序,默认效果如图 4-2(a)所示,单击界面中的"显示带按钮的对话框"按钮,效果如图 4-2(b)所示,单击界面中的"显示列表的对话框"按钮,效果如图 4-2(c)所示,单击界面中的"显示带单选列表项的对话框"按钮,效果如图 4-2(d)所示,单击界面中的"显示带多选列表项的对话框"按钮,效果如图 4-2(e)所示,单击图 4-2(c)中对话框中的"确定"按钮,效果如图 4-2(f)所示。

前面两个例子中介绍了几种常用的 AlertDialog 对话框,下面通过综合介绍一种用户登录对话框。登录对话框中界面中包含 3 个输入框及 3 个按钮。其操作步骤为:

(1) 在 Eclipse 环境下建立 AlertDialog_Cutom 的工程。

(2) 编写布局文件,布局一个按钮控件。打开 res/Layout 目录下的 main.xml 文件,其代码修改为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
```



(a) 默认界面



(b) 带按钮对话框



(c) 带列表项对话框



(d) 带单选列表项的对话框



(e) 带多选列表项的对话框



(f) 显示所选项

图 4-2　各种类型的对话框

```
    android:layout_height = "fill_parent"
    android:gravity = "center_horizontal">
<Button
    android:id = "@ + id/bn"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "登录对话框"/>
</LinearLayout>
```

(3) 创建布局文件,布局 3 个编辑框、4 个文本框及 1 个按钮控件。在打开 res/Layout 目录下新建 denglu.xml 文件。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<TableLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@ + id/loginForm"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
<TableRow>
<TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "用户名:"
    android:textSize = "10pt"     />
<! -- 输入用户名的文本框 -->
<EditText
```

```
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:hint = "请填写登录账号"
        android:selectAllOnFocus = "true"    />
</TableRow>
<TableRow>
<TextView
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "密码:"
        android:textSize = "10pt"    />
<!-- 输入密码的文本框 -->
<EditText
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:password = "true"
        android:hint = " ****** "
        android:textSize = "10pt"    />
</TableRow>
<TableRow>
<TextView
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "电话号码:"
        android:textSize = "10pt"    />
<!-- 输入电话号码的文本框 -->
<EditText
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:hint = "请输入电话号码"
        android:selectAllOnFocus = "true"
        android:phoneNumber = "true"    />
</TableRow>
<Button
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "注册"/>
</TableLayout>
```

(4) 编写 Activity 文件,实现自定义的对话框。打开 src/com.example.alertdialog_cutom 包下的 MainActivity.java。代码为:

```
package com.example. alertdialog_cutom;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.AlertDialog.Builder;
import android.content.DialogInterface;
import android.content.DialogInterface.OnClickListener;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
```

```java
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TableLayout;
public class MainActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button bn = (Button)findViewById(R.id.bn);
        //定义一个 AlertDialog.Builder 对象
        final Builder builder = new AlertDialog.Builder(this);
        //为按钮绑定事件监听器
        bn.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View source)
            {
                //设置对话框的图标
                builder.setIcon(R.drawable.sm);
                //设置对话框的标题
                builder.setTitle("自定义普通对话框");
                //装载/res/layout/login.xml 界面布局
                TableLayout loginForm = (TableLayout)getLayoutInflater()
                    .inflate(R.layout.denglu,null);
                //设置对话框显示的 View 对象
                builder.setView(loginForm);
                //为对话框设置一个"确定"按钮
                builder.setPositiveButton("登录"
                    //为按钮设置监听器
                    ,new OnClickListener()
                    {
                        @Override
                        public void onClick(DialogInterface dialog,int which)
                        {
                            //此处可执行登录处理
                        }
                    });
                //为对话框设置一个"取消"按钮
                builder.setNegativeButton("取消",new OnClickListener()
                    {
                        @Override
                        public void onClick(DialogInterface dialog,int which)
                        {
                            //取消登录,不做任何事情
                        }
                    });
                //创建、并显示对话框
                builder.create().show();
```

```
                }
            });
        }
    }
```


运行程序，在登录界面中单击"登录"按钮，得到如图 4-3 所示。



图 4-3　登录对话框界面

4.2　Android 提示框

在 Android 项目开发中，经常需要将一些临时信息显示给用户，虽然使用前面介绍的第 3 章介绍的基本控件可以达到显示信息的目的，但是这样做不仅会增加代码量，而且对于用户来说也不够友好。为此，Android 提供了消息提示框与对话框来显示这些信息。

4.2.1　Android 消息提示框

在前面的实例中，已经应用过 Toast 类来显示一个简单的消息提示框了。本节将对 Toast 进行详细介绍。Toast 类用于在屏幕中显示一个消息提示框，该消息提示框没有任何控制按钮，并且不会获得焦点，经过一定时间后自动消失。通常用于显示一些快速提示信息，应用范围非常广泛。

使用 Toast 来显示消息提示框比较简单，只需要经过以下 3 个步骤即可实现。

① 创建一个 Toast 对象。通常有两种方法：一种是使用构造方式进行创建；另一种是调用 Toast 类的 makeText()方法创建。

使用构造方法创建一个名称为 toast 的 Toast 类对象的代码为：

```
Toast toast = new Toast(this);
```

调用 Toast 类的 makeText()方法创建一个名称为 toast 的 Toast 对象的代码为：

```
Toast toast = Toast.makeText(this,"要显示的内容",Toast.LENGTH_SHORT);
```

② 调用 Toast 类提供的方法来设置该消息提示框的对齐方式、页边距、显示等内容。常用方法如表 4-3 所示。

表 4-3　Toast 类的常用方法

方　　法	描　　述
setDuration(int duration)	用于设置消息提示框持续的时间，参数值通常使用 Toat.LENGTH_LONG 或 Toast.LENGTH_SHORT
setGravity(int gravity, int xOffset, int yOffset)	用于设置消息提示框的位置，参数 gravity 用于指定对齐方式；xOffset 和 yOffset 用于指定具体的偏移值
setMargin(float horizontalMargin, float verticalMargin)	用于设置消息提示的页边距
setText(CharSequence s)	用于设置要显示的文本内容
setView(View view)	用于设置将要在消息提示框中显示的视图

③ 调用 Toast 类的 show()方法显示消息提示框。值得注意的是，一定要调用该方法，否则设置的消息提示框将不显示。

下面通过一个具体实例来演示怎样使用 Toast 类显示消息提示框。

【例 4-3】 实现几种类型的 Toast 类。其实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Toast_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明两个线性布局、两个 TextView 控件及一个 ImageView 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout
  xmlns:android = "http://schemas.android.com/apk/res/android"
  android:layout_height = "wrap_content" android:layout_width = "wrap_content"
  android:background = "#ffffffff" android:orientation = "vertical"
  android:id = "@ + id/llToast" >
  <TextView
    android:layout_height = "wrap_content"
    android:layout_margin = "1dip"
    android:textColor = "#ffffffff"
    android:layout_width = "fill_parent"
    android:gravity = "center"
    android:background = "#bb000000"
    android:id = "@ + id/tvTitleToast" />
  <LinearLayout
    android:layout_height = "wrap_content"
    android:orientation = "vertical"
    android:id = "@ + id/llToastContent"
```

```
        android:layout_marginLeft = "1dip"
        android:layout_marginRight = "1dip"
        android:layout_marginBottom = "1dip"
        android:layout_width = "wrap_content"
        android:padding = "15dip"
        android:background = "#44000000" >
        < ImageView
         android:layout_height = "wrap_content"
         android:layout_gravity = "center"
         android:layout_width = "wrap_content"
         android:id = "@ + id/tvImageToast" />
        < TextView
         android:layout_height = "wrap_content"
         android:paddingRight = "10dip"
         android:paddingLeft = "10dip"
         android:layout_width = "wrap_content"
         android:gravity = "center"
         android:textColor = "#ff000000"
         android:id = "@ + id/tvTextToast" />
      </LinearLayout >
    </LinearLayout >
```

(3) 在 res\layout 目录下新建一个 custom.xml 文件，该文件用于实现弹出对话的消息提示框界面。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout
  xmlns:android = "http://schemas.android.com/apk/res/android"
  android:layout_height = "wrap_content" android:layout_width = "wrap_content"
  android:background = "#ffffffff" android:orientation = "vertical"
  android:id = "@ + id/llToast" >
  < TextView
    android:layout_height = "wrap_content"
    android:layout_margin = "1dip"
    android:textColor = "#ffffffff"
    android:layout_width = "fill_parent"
    android:gravity = "center"
    android:background = "#bb000000"
    android:id = "@ + id/tvTitleToast" />
  < LinearLayout
    android:layout_height = "wrap_content"
    android:orientation = "vertical"
    android:id = "@ + id/llToastContent"
    android:layout_marginLeft = "1dip"
    android:layout_marginRight = "1dip"
    android:layout_marginBottom = "1dip"
    android:layout_width = "wrap_content"
    android:padding = "15dip"
    android:background = "#44000000" >
    < ImageView
     android:layout_height = "wrap_content"
```

```
        android:layout_gravity = "center"
        android:layout_width = "wrap_content"
        android:id = "@ + id/tvImageToast" />
      < TextView
        android:layout_height = "wrap_content"
        android:paddingRight = "10dip"
        android:paddingLeft = "10dip"
        android:layout_width = "wrap_content"
        android:gravity = "center"
        android:textColor = " # ff000000"
        android:id = "@ + id/tvTextToast" />
  </LinearLayout >
</LinearLayout >
```

（4）打开 src\fs. toast_test 包下的 MainActivity. java 文件，在文件中实现当单击界面中的某个按钮时，即弹出几种对应的 Toast 提示框。代码为：

```
package fs.toast_test;
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.View.OnClickListener;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;
public class MainActivity extends Activity implements OnClickListener {
 Handler handler  =  new Handler();
 @Override
 public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  findViewById(R.id.btnSimpleToast).setOnClickListener(this);
  findViewById(R.id.btnSimpleToastWithCustomPosition).setOnClickListener(this);
  findViewById(R.id.btnSimpleToastWithImage).setOnClickListener(this);
  findViewById(R.id.btnCustomToast).setOnClickListener(this);
  findViewById(R.id.btnRunToastFromOtherThread).setOnClickListener(this);
}
public void showToast() {
 handler.post(new Runnable() {
    @Override
    public void run() {
    Toast.makeText(getApplicationContext(),"我来自其他线程!",
      Toast.LENGTH_SHORT).show();
    }
 });
}
```

```
@Override
public void onClick(View v) {
 Toast toast = null;
 switch (v.getId()) {
 case R.id.btnSimpleToast:
    Toast.makeText(getApplicationContext(),"默认 Toast 样式",
      Toast.LENGTH_SHORT).show();
    break;
 case R.id.btnSimpleToastWithCustomPosition:
    toast = Toast.makeText(getApplicationContext(),
      "自定义位置 Toast",Toast.LENGTH_LONG);
    toast.setGravity(Gravity.CENTER,0,0);
    toast.show();
    break;
 case R.id.btnSimpleToastWithImage:
    toast = Toast.makeText(getApplicationContext(),
      "带图片的 Toast",Toast.LENGTH_LONG);
    toast.setGravity(Gravity.CENTER,0,0);
    LinearLayout toastView = (LinearLayout) toast.getView();
    ImageView imageCodeProject = new ImageView(getApplicationContext());
    imageCodeProject.setImageResource(R.drawable.g4);
    toastView.addView(imageCodeProject,0);
    toast.show();
    break;
 case R.id.btnCustomToast:
    LayoutInflater inflater = getLayoutInflater();
    View layout = inflater.inflate(R.layout.custom,
      (ViewGroup) findViewById(R.id.llToast));
    ImageView image = (ImageView) layout
      .findViewById(R.id.tvImageToast);
    image.setImageResource(R.drawable.g4);
    TextView title = (TextView) layout.findViewById(R.id.tvTitleToast);
    title.setText("Attention");
    TextView text = (TextView) layout.findViewById(R.id.tvTextToast);
    text.setText("完全自定义 Toast");
    toast = new Toast(getApplicationContext());
    toast.setGravity(Gravity.RIGHT | Gravity.TOP,12,40);
    toast.setDuration(Toast.LENGTH_LONG);
    toast.setView(layout);
    toast.show();
    break;
 case R.id.btnRunToastFromOtherThread:
    new Thread(new Runnable() {
    public void run() {
      showToast();
    }
    }).start();
    break;
  }
 }
}
```

运行程序，默认效果如图 4-4(a)所示；单击界面中的“默认”按钮，效果如图 4-4(b)所示；单击界面中的“自定义显示位置”按钮，效果如图 4-4(c)所示；单击界面中的“带图片”

按钮，效果如图 4-4(d)所示；单击界面中的“完全自定义”按钮，效果如图 4-4(e)所示；单击界面中的“其他线程”按钮，效果如图 4-4(f)所示。



(a) 默认界面　(b) 默认Toast样式　(c) 自定义位置Toast样式

(d) 带图片的Toast样式　(e) 完全自定义Toast样式　(f) 其他线程Toast样式

图 4-4　Toast 各种样式

4.2.2　Android 状态栏上的通知

在 Android 系统中，应用程序可能会遇到几种情况需要通知用户，有的需要用户回应，有的则不需要，例如：

- 当保存文件等事件完成，应该会出现一个小的消息，以确认保存成功。
- 如果应用程序在后台运行，需要用户的注意，应用程序应该创建一个通知，允许用户为他或她的回应提供便利
- 如果应用程序正在执行工作，用户必须等待（如装载文件），应用程序应该显示进度或等待提醒。

针对这些情况，Android 都提供了不同的提醒方式。主要包括下面几种：

(1) Toast Notification：是指出现在屏幕上的暂时性通知，这种通知用于传达一些告知类型的消息，短暂停留后会自动消失，无须用户交互。例如告知下载已完成等。

(2) Status Bar Notification：是指以一个图标或者滚动条文本的形式出现在系统顶部状态栏上的通知。当应用程序处于后台运行状态时，这种方式比较合适。这种通知形式的好处是既能被关注到，又无须打断当前任务，可以从顶部下拉查看通知并做选择性处理。

(3) Dialog Notification：类似于 iOS 的 Alert Notification，以对话窗口的形式出现在屏幕上，用于重要或需及时处理的通知。

下面通过一个实例来说明怎样使用 Notification 在状态栏上显示通知。

【例 4-4】 实现在状态栏上显示通知和删除通知。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Notification_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明两个 Button 控件，一个用于显示通知，另一个用于删除通知。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "#aabbcc">
    < Button
        android:id = "@ + id/button1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "显示通知" />
    < Button
        android:id = "@ + id/button2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "删除通知" />
</LinearLayout >
```

(3) 在 res\layout 目录下新建一个 content.xml 文件，在文件中声明一个 TextView 控件，用于显示通知内容界面。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:orientation = "vertical" >
    < TextView
        android:id = "@ + id/textView1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "晚上 19: 00 电视新闻联播"
        android:textAppearance = "?android:attr/textAppearanceMedium" />
</LinearLayout >
```

(4) 打开 src\fs.notification_test 包下的 MainActivity.java 文件，在文件中实现通知的显示及通知删除。代码为：

```
package fs.notification_test;
import android.app.Activity;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Bundle;
```

```
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
public class MainActivity extends Activity {
    final int NOTIFYID_1 = 123;                              //第 1 个通知的 ID
    final int NOTIFYID_2 = 124;                              //第 2 个通知的 ID
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //获取通知管理器,用于发送通知
         final NotificationManager notificationManager = (NotificationManager) getSystemService
(NOTIFICATION_SERVICE);
        Button button1 = (Button) findViewById(R.id.button1);  //获取"显示通知"按钮
        //为"显示通知"按钮添加单击事件监听器
        button1.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                Notification notify = new Notification();    //创建一个 Notification 对象
                notify.icon = R.drawable.ic_launcher;
                notify.tickerText = "显示第 1 个通知";
                notify.when = System.currentTimeMillis();    //设置发送时间
              //设置默认声音、默认振动和默认闪光灯
                notify.defaults = Notification.DEFAULT_ALL;
            //设置事件信息
        notify.setLatestEventInfo(MainActivity.this,"无题","每天进步一点点",   null);
                                                            //通过通知管理器发送通知
        notificationManager.notify(NOTIFYID_1,notify);
                //添加第 2 个通知
                Notification notify1 = new Notification(R.drawable.g4,
                        "显示第 3 个通知",System.currentTimeMillis());
                notify1.flags| = Notification.FLAG_AUTO_CANCEL;
          //打开应用程序后图标消失
                Intent intent = new Intent(MainActivity.this,ContentActivity.class);
                PendingIntent pendingIntent = PendingIntent.getActivity(MainActivity.this,
0,intent,0);
                notify1.setLatestEventInfo(MainActivity.this,"通知",
                        "查看详细内容",pendingIntent);       //设置事件信息
        notificationManager.notify(NOTIFYID_2,notify1);      //通过通知管理器发送通知
            }
        });
        Button button2 = (Button) findViewById(R.id.button2);
                                  //获取"删除通知"按钮为"删除通知"按钮添加单击事件监听器
        button2.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
//notificationManager.cancel(NOTIFYID_1);             //清除 ID 号为常量 NOTIFYID_1 的通知
                notificationManager.cancelAll();             //清除全部通知
            }
        });
    }
```

```
}
```

(5) 在 src\fs. notification_test 包下新建一个 ContentActivity. java 文件，用于实现调用 conten. xml 文件。代码为：

```
package fs.notification_test;
import android.app.Activity;
import android.os.Bundle;
public class ContentActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //TODO 自动存根法
        super.onCreate(savedInstanceState);
        setContentView(R.layout.content);
    }
}
```

(6) 打开 AndroidManifest. xml 文件，在文件中为第 1 个通知设置默认声音、默认振动和默认闪光灯；并且在程序中还需要启动另一个活动 ContentActivity，即声明 Activity。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.notification_test"
    android:versionCode = "1"
    android:versionName = "1.0" >
    < uses - sdk android:minSdkVersion = "15" />
    <! -- 添加操作闪光灯的权限 -->
    < uses - permission android:name = "android.permission.FLASHLIGHT" />
    <! -- 添加操作振动器的权限 -->
    < uses - permission android:name = "android.permission.VIBRATE" />
    < application
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        < activity
            android:name = "fs.notification_test.MainActivity"
            android:label = "@string/app_name" >
            < intent - filter >
                < action android:name = "android.intent.action.MAIN" />
                < category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter >
        </activity >
          < activity
            android:label = "详细内容"
            android:name = ".ContentActivity"
            android:theme = "@android:style/Theme.Dialog" />
    </application >
```

```
</manifest>
```

运行程序,默认效果如图 4-5(a)所示;单击界面中的“显示通知”按钮,效果如图 4-5(b)、图 4-5(c)及图 4-5(d)所示;单击界面中的“删除通知”按钮,效果如图 4-5(e)所示。



(a) 默认界面　(b) 状态栏上显示通知1　(c) 状态栏上显示通知2

(d) 显示通知图标　(e) 删除通知效果

图 4-5　显示通知

4.3　Android 闹钟设置

AlarmManager 类是 Android 提供的用于在未来的指定时间弹出一个警告信息或完成指定操作的类。实际上,AlarmManager 是一个全局的定时器,使用它可以在指定的时间或周期启动其他的组件(包括 Activity、Service 和 BroadcastReceiver)。使用 AlarmManager 设置警告后,Android 将自动开启目标应用,即使手机处于休眠状态。因此,使用 AlarmManager 也可以实现关机后仍可以响应的闹钟。

在 Android 中,要获取 AlarmManager 对象,类似于获取 NotificationManager 服务,也需要使用 Context 类的 getSystemService()方法来实现。实现代码为:

```
Context.getSystemService(Context.ALARM_SERVICE)
```

获取 AlarmManager 对象后,就可以应用该对象提供的相关方法来设置警告。AlarmManager 对象提供的常用方法如表 4-4 所示。

表 4-4 AlarmManager 对象提供的常用方法

方　　法	描　　述
cancel(PendingIntent operation)	用于取消已经设置的与参数匹配的闹钟
set(int type,long triggerAtTime,PendingIntent operation)	用于设置一个新的闹钟
setInexactRepeating(int type, long triggerAtTime, long interval,PendingIntent operation)	用于设置一个非精确重复类型的闹钟。例如,设置一个每小时启动一次的闹钟,但是系统并不一定总在每个小时的开始启动闹钟
setRepeating(int type, long triggerAtTime, long interval,PendingIntent operation)	用于设置一个重复类型的闹钟
setTime(long millis)	用于设置闹钟的时间
setTimeZone(String timeZone)	用于设置系统默认的时区

在设置闹钟时,AlarmManager 提供了以下 4 种类型。

- ELAPSED_REALTIME:用于设置从现在开始过一定时间后启动的闹钟。当系统进入睡眠状态时,这种类型的闹钟不会唤醒系统,直到系统下次被唤醒才传递它,该闹钟所用的时间是相对时间,是从系统启动后开始计时的(包括睡眠时间),可以通过调用 SystemClock.elapsedRealtime()方法获得。
- ELAPSED_REALTIME_WAKEUP:用于设置从现在开始过一定时间后启动的闹钟。这种类型的闹钟能够唤醒系统,使用方法与 ELAPSED_REALTIME 类似,也可以通过调用 SystemClock.elapsedRealtime()方法获得。
- RTC:用于设置当系统调用 System.currenTimeMillis()方法的返回值与指定的触发时间相等时启动的闹钟。当系统进入睡眠状态时,这种类型的闹钟不会唤醒系统,直到系统下次被唤醒才传递它,该闹钟所用的时间是绝对时间、UTC 时间,可以通过调用 System.currentTimeMillis()方法获得。
- RTC_WAKEUP:用于设置当系统调用 System.currentTimeMillis()方法的返回值与指定的触发时间相等时启动的闹钟。这种类型的闹钟能够唤醒系统,使用方法与 RTC 类似。

使用 AlarmManager 设置闹钟比较简单,下面通过一个实例来介绍怎样设计一个闹钟。

【例 4-5】 使用 AlarmManager 实现一个简单的警告服务。其实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 AlarmManager_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件,在文件中声明一个 TextView 控件及两个 Button 控件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = " # aabbcc">
< TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
```

```
        android:text = "警告服务"/>
    < Button
        android:id = "@ + id/startalarm"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "开始"/>
    < Button
        android:id = "@ + id/cancelalarm"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "取消"/>
    </LinearLayout >
```

(3) 打开 src\fs. alarmactivity_test 包下的 MainActivity. java 文件,在文件中实现闹钟警告。代码为:

```
    package fs. alarmmanager_test;
    import java. util. Calendar;
    import android. app. Activity;
    import android. app. AlarmManager;
    import android. app. PendingIntent;
    import android. content. Intent;
    import android. os. Bundle;
    import android. os. SystemClock;
    import android. view. View;
    import android. widget. Button;
    import android. widget. Toast;
    public class MainActivity extends Activity {
        private PendingIntent pendingIntent;
        /** 第 1 次调用 activity 活动. */
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super. onCreate(savedInstanceState);
            setContentView(R. layout. main);
            Button buttonStart = (Button)findViewById(R. id. startalarm);
            Button buttonCancel = (Button)findViewById(R. id. cancelalarm);
            buttonStart. setOnClickListener(new Button. OnClickListener(){
                @Override
                public void onClick(View arg0) {
                    //创建一个 Intent 对象
                    Intent myIntent = new Intent(MainActivity. this,AlarmService. class);
            //显示闹钟的 PendingIntent 对象
            pendingIntent = PendingIntent. getService(MainActivity. this,0,myIntent,0);
            //获取 AlarmManager 对象
                    AlarmManager alarmManager = (AlarmManager)getSystemService(ALARM_SERVICE);
                    Calendar calendar = Calendar. getInstance();
    //设置闹钟的分钟数
                    calendar. setTimeInMillis(System. currentTimeMillis());
                    calendar. add(Calendar. SECOND,10);
                    alarmManager. set (AlarmManager. RTC _ WAKEUP, calendar. getTimeInMillis (),
    pendingIntent);
```

```
    //设置一个闹钟,并显示一个消息提示
        Toast.makeText(MainActivity.this,"开始警告",Toast.LENGTH_LONG).show();
                }});
            buttonCancel.setOnClickListener(new Button.OnClickListener(){
                @Override
                public void onClick(View arg0) {
                    //TODO 自动存根法
                    AlarmManager alarmManager = (AlarmManager)getSystemService(ALARM_SERVICE);
                    alarmManager.cancel(pendingIntent);
        Toast.makeText(MainActivity.this,"取消!",Toast.LENGTH_LONG).show();
                    }});
            }
        }
```

（4）在 src\fs.alarmactivity_test 包下创建一个 AlarmService.java 文件，在文件中实现闹钟警告显示。代码为：

```
package fs.alarmmanager_test;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.widget.Toast;
public class AlarmService extends Service {
    @Override
    public void onCreate() {
        //TODO 自动存根法
        Toast.makeText(this,"创建闹钟警告",Toast.LENGTH_LONG).show();
    }
    @Override
    public IBinder onBind(Intent intent) {
        //TODO 自动存根法
        Toast.makeText(this,"绑定闹钟警告",Toast.LENGTH_LONG).show();
        return null;
    }
    @Override
    public void onDestroy() {
        //TODO 自动存根法
        super.onDestroy();
        Toast.makeText(this,"删除闹钟警告",Toast.LENGTH_LONG).show();
    }
    @Override
    public void onStart(Intent intent,int startId) {
        //TODO 自动存根法
        super.onStart(intent,startId);
        Toast.makeText(this,"开始闹钟警告",Toast.LENGTH_LONG).show();
    }
    @Override
    public boolean onUnbind(Intent intent) {
        //TODO 自动存根法
        Toast.makeText(this,"不绑定闹钟警告",Toast.LENGTH_LONG).show();
        return super.onUnbind(intent);
```

```
        }
    }
```

(5) 打开 AndroidManifest.xml 文件中配置 AlarmActivity,配置主要属性 Activity 使用的实现。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.alarmmanager_test"
    android:versionCode = "1"
    android:versionName = "1.0" >
    < uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
    < application
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        < activity
            android:name = "fs.alarmmanager_test.MainActivity"
            android:label = "@string/app_name" >
            < intent - filter >
                < action android:name = "android.intent.action.MAIN" />
                < category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter >
        </activity >
        <! -- 设置闹钟警告权限 -->
      < service android:name = ".AlarmService" />
    </application >
</manifest >
```

运行程序,默认效果如图 4-6(a)所示,单击界面中的"开始"按钮,即实现闹钟警告,如图 4-6(b)所示,单击界面中的"取消"按钮,即取消闹钟警告,效果如图 4-6(c)所示。



(a) 默认界面　　(b) 闹钟警告　　(c) 取消闹钟警告

图 4-6　警告

4.4 Android 菜单

在部分的应用程序都包括两种人机互动方式，一种是直接通过 GUI 的 Views，其可以满足大部分的交互操作；另外一种是应用 Menu，当按下 Menu 按钮后，会弹出与当前活动状态下的应用程序相匹配的菜单。这两种方式相比较都有各自的优势，而且可以很好地相辅相成，即便用户可以由主界面完成大部分操作，但是适当地拓展 Menu 功能可以更加完善应用程序，至少用户可以通过排列整齐的按钮清晰地了解当前模式下可以使用的功能。

Android 提供了 3 种菜单类型，分别为 Options Menu、Context Menu、Sub Menu。

Options Menu：该菜单为选项菜单，是通过＜home＞键来显示，Options Menu 需要在 View 上按上 2s 后显示。这两种 Menu 都可以加入子菜单，子菜单不能嵌套子菜单。Options Menu 最多只能在屏幕最下面显示 6 个菜单选项，称为 Icon Menu，Icon Menu 不能有 Checkable 选项。多于 6 个的菜单项会以 More Icon Menu 来调出，称为 Expanded Menu。Options Menu 通过 Activity 的 onCreateOptionsMenu 来生成，这个函数只会在 Menu 第 1 次生成时调用。任何想改变 Options Menu 的想法只能在 onPrepareOptionsMenu 来实现，这个函数会在 Menu 显示前调用。onOptionsItemSelected 用来处理选中的菜单项。

Context Menu：该菜单为上下文菜单，是跟某个具体的 View 绑定在一起的，在 Activity 中用 Register ForContextMenu 来为某个 View 注册 Context Menu。Context Menu 在显示前都会调用 onCreateContextMenu 来生成 Menu。onContextItemSelected 用来处理选中的菜单项。

Android 还提供了对菜单项进行分组的功能，可以把相似功能的菜单项分在同一个组，这样就可以通过调用 setGroupCheckable、setGroupEnabled、setGroupVisiable 来设置菜单属性，而无须单独设置。

4.4.1 Android 选项菜单

当 Activity 在前台运行时，如果用户按下手机上的 Menu 键，此时就会在屏幕底端弹出相应的选项菜单。但这个功能是需要开发人员编程来实现的，如果在开发应用程序时没有实现该功能，那么程序运行时按下手机上的 Menu 键是不会起作用的。

对于携带图标的选项菜单，每次最多只能显示 6 个，当菜单选项多于 6 个时，将只显示前 5 个和 1 个扩展菜单选项，单击扩展菜单选项将弹出其余菜单项。扩展菜单项中将不会显示图标，但是可以显示单选框及复选框。

在 Android 中通过回调方法来创建菜单并处理菜单项按下的事件，这些回调方法及说明如表 4-5 所示。

表 4-5　选项菜单相关的回调方法及说明

方　法　名	描　　述
onCreateOptionsMenu(Menu menu)	初始化选项菜单，该方法只在第 1 次显示菜单时调用，如果需要每次显示菜单时更新菜单项，则需要重写 onPrepareOptionsMenu(Menu)方法
public boolean onOptionsItemSelected(MenuItem item)	当选项菜单中某个选项被选中时调用该方法，默认是一个返回 false 的空实现

续表

方　法　名	描　　述
public void onOptionsMenu(Menu menu)	当选项菜单关闭时(或由于用户按下了返回键或是选择了某个菜单项)调用该方法
public boolean onPrepareOptionsMenu (Menu menu)	为程序准备选项菜单,每次选项菜单显示前会调用该方法。可以通过该方法设置某些菜单项可用、不可用或修改菜单项的内容。重写该方法时需要返回true,否则选项菜单将不会显示

提示:除了使用开发回调方法 onOptionsItemSelected 来处理用户选中的菜单事件,还可以为每个菜单项 MenuItem 对象添加 OnMenuItemClickListener 监听器来处理菜单项选中事件。

开发选项菜单将主要用到 Menu、MenuItem 及 SubMenu,下面对它们进行简单介绍。

1. Menu 类

一个 Menu 对象代表一个菜单,在 Menu 对象中可以添加菜单项 MenuItem,也可以添加子菜单 SubMenu。在 Menu 中常用的方法如表 4-6 所示。

表 4-6　Menu 的常用方法及说明

方法名称	参数说明	方法说明
• MenuItem add(int groupId,int itemId,int order,CharSequence title); • MenuItem add(int groupId,int itemId,int order,int titleRes); • MenuItem add(CharSequence title); • MenuItem add(int titleRes)	groupId 为菜单项所在的组 id,通过分组可以对菜单项进行批量操作,如果菜单项不需要属于任何组,则传入 NONE; itemId 为唯一标识菜单项的 id,可传入 NONE; order 为菜单项的顺序,可以传入 NONE; title 为菜单项显示的文本内容; titleRes 为 String 对象的资源标识符	向 Menu 添加 1 个菜单项,返回 MenuItem 对象
• SubMenu addSubMenu(int titleRes); • SubMenu addSubMenu(int groupId,int itemId,int order,int titleRes); • SubMenu addSubMenu(CharSequence title); • SubMenu addSubMenu(int groupId,int itemId,int order,CharSequence title)	groupId 为子菜单所在的组 id,通过分组可以对子菜单进行批量操作,如果子菜单不需要属于任何组,则传入 NONE; itemId 为唯一标识子菜单的 id,可传入 NONE; order 为子菜单的顺序,可传入 NONE; title 为子菜单显示的文本内容; titleRes 为 String 对象的资源标识符	向 Menu 添加 1 个子菜单,返回 SubMenu 对象
void clear()	----	移除菜单中所有的子项
void close()	---	如果菜单正在显示,则关闭菜单
MenuItem findItem(int id)	id:MenuItem 的标识符	返回指定 id 的 MenuItem 对象
void removeGroup(int groupId)	groupId 为组 id	如果指定 id 的组不为空,则从菜单中移除该组
void removeItem(int id)	id 为 MenuItem 的 id	移除指定 id 的 MenuItem
int size()	---	返回 Menu 中菜单项的个数

2. MenuItem 对象

MenuItem 对象代表一个菜单项，通常 MenuItem 实例通过 Menu 的 add 方法获得，在 MenuItem 中常用的成员方法如表 4-7 所示。

表 4-7 选项菜单相关的回调方法及说明

方法名称	参数说明	方法说明
setAlphabeticShortcut(char alphaChar)	alphaChar 为字母快捷键	设置 MenuItem 的字母快捷键
MenuItem setNumericShortcut（char numericChar）	numericChar 为数字快捷键	设置 MenuItem 的数字快捷键
MenuItem setIcon(Drawable icon)	icon 为图标 Drawable 对象	设置 MenuItem 的图标
MenuItem setIntent(Intent intent)	intent 为与 MenuItem 绑定的 Intent 对象	为 MenuItem 绑定 Intent 对象，当被选中时，将会调用 startActivity 方法处理相应的 Intent
setOnMenuItemClickListener (MenuItem. OnMenuItemClickListener menuItemClickListener)	menuItemClickListener 为监听器	为 MenuItem 设置自定义的监听器，一般情况下，使用回调方法 onOptionsItemSelected 会更有效率
setShortcut（char numericChar, char alphaChar）	numericChar 为数字快捷键；alphaChar 为字母快捷键	为 MenuItem 设置数字快捷键和字母快捷键，当按下快捷键或按住 Alt 的同时按下快捷键时将会触发 MenuItem 的选中事件
setTitle(int title)	title 为标题的资源 id	为 MenuItem 设置标题
setTitle(CharSequence title)	title 为标题的名称	
setTitleCondensed(CharSequence title)	title 为 MenuItem 的缩略标题	设置 MenuItem 的缩略标题，当 MenuItem 不能显示全部的标题时，将显示缩略标题

3. SubMenu 对象

SubMenu 继承自 Menu，每个 SubMenu 实例代表一个子菜单，SubMenu 中常用的方法如表 4-8 所示。

表 4-8 SubMenu 中常用方法

方法名称	参数说明	方法说明
setHeaderIcon(Drawable icon)	icon 为标题图标 Drawable 对象	设置子菜单的标题图标
setHeaderIcon(int iconRes)	iconRes 为标题图标的资源 id	
setHeaderTitle(int titleRes)	titleRes 为标题文本的资源 id	设置子菜单的标题
setHeaderTitle(CharSequence title)	title 为标题文本对象	
setIcon(Drawable icon)	icon 为图标 Drawable 对象	设置子菜单在父菜单中显示的图标
setIcon(int iconRes)	iconRes 为图标资源 id	
setHeaderView(View view)	view 为用于子菜单标题的 View 对象	设置指定的 View 对象为子菜单图标

下面通过一个实例来演示怎样使用 Options Menu。

【例 4-6】 实现一个蓝牙连接功能。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 OptionsMenu_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明一个 TextView 控件。代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity" >
      <TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_centerHorizontal = "true"
        android:layout_centerVertical = "true"
        android:text = "@string/hello_world" />
</RelativeLayout>
```

(3) 打开 res\menu 目录下的 main.xml 文件，在文件中实现菜单选项的声明。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<menu xmlns:android = "http://schemas.android.com/apk/res/android">
    <item android:id = "@ + id/connect"
        android:orderInCategory = "100"
        android:showAsAction = "never"
        android:icon = "@android:drawable/ic_menu_send"
        android:title = "连接" />
    <item android:id = "@ + id/disconnect"
        android:orderInCategory = "100"
        android:showAsAction = "never"
        android:icon = "@android:drawable/ic_menu_close_clear_cancel"
        android:title = "断开" />
    <item android:id = "@ + id/search"
        android:orderInCategory = "100"
        android:showAsAction = "never"
        android:icon = "@android:drawable/ic_menu_search"
        android:title = "发现" />
    <item android:id = "@ + id/view"
        android:orderInCategory = "100"
        android:showAsAction = "never"
        android:icon = "@android:drawable/ic_menu_view"
        android:title = "查看" />
    <item android:id = "@ + id/help"
        android:orderInCategory = "100"
        android:showAsAction = "never"
        android:icon = "@android:drawable/ic_menu_help"
        android:title = "帮助" />
    <item android:id = "@ + id/exit"
        android:orderInCategory = "100"
```

```
        android:showAsAction="never"
        android:icon="@android:drawable/ic_menu_revert"
        android:title="退出" />
</menu>
```


(4) 打开 src/optionsmenu_test 包下的 MainActivity.java 文件,在文件实现菜单选项,并当单击界面中的选项时,即弹出对应的消息提示框。代码为:

```
package fs.optionsmenu_test;
import android.os.Bundle;
import android.app.Activity;
import android.widget.Toast;
import android.view.Menu;
import android.view.MenuItem;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    //初始化菜单
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        //加载菜单
        getMenuInflater().inflate(R.menu.main,menu);
        return true;
    }
    //根据菜单执行相应内容
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
        case R.id.connect:
            Toast.makeText(getApplicationContext(),"蓝牙连接……",Toast.LENGTH_SHORT).show();
            return true;
        case R.id.disconnect:
            Toast.makeText(getApplicationContext(),"蓝牙断开……",Toast.LENGTH_SHORT).show();
            return true;
        case R.id.search:
            Toast.makeText(getApplicationContext(),"寻找蓝牙……",Toast.LENGTH_SHORT).show();
            return true;
        case R.id.view:
Toast.makeText(getApplicationContext(),"查看……",Toast.LENGTH_SHORT).show();
            return true;
        case R.id.help:
Toast.makeText(getApplicationContext(),"帮助……",Toast.LENGTH_SHORT).show();
            return true;
        case R.id.exit:
Toast.makeText(getApplicationContext(),"退出……",Toast.LENGTH_SHORT).show();
            return true;
        }
        return false;
```

```
    }
}
```

运行程序，默认效果如图 4-7(a)所示，当单击界面中右侧的 Menu 选项时，效果如图 4-7(b)所示，当选择对应的菜单项时，效果如图 4-7(c)所示。



(a) 默认界面　(b) 菜单项界面　(c) 消息提示框界面

图 4-7　菜单项 1

当需要将界面设置成“老”样式界面时，需要改动的并不大，仅需要修改 AndroidManifest.xml 文件的样式即可。

```
…
        <activity
            android:name = "fs.optionsmenu_test.MainActivity"
            android:label = "@string/app_name"
            android:theme = "@android:style/Theme" >
            <intent - filter>
                <action android:name = "android.intent.action.MAIN" />
                <category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter>
        </activity>
    </application>
</manifest>
```

重新运行程序，默认效果如图 4-8(a)所示，当单击界面中右侧的 Menu 选项时，效果如图 4-8(b)所示，当选择对应的菜单项时，效果如图 4-8(c)所示。



(a) 默认界面　(b) “老”样式菜单项界面　(c) 消息提示框界面

图 4-8　菜单项 2

4.4.2 Android 子菜单

在 Android 中除了使用菜单之外，还可以设置子菜单(SubMenu)。通过子菜单可以将相同类别的菜单命令归类，带来更好的用户体验。SubMenu 类便提供了子菜单的一些操作。SubMenu 类继承于 Menu 类，每一个 SubMenu 对象代表了一个子菜单。

SubMenu 菜单的主要方法有：

- setIcon(int iconRes)：用于设置子菜单显示的图标。
- add(int titleRes)：向子菜单中添加菜单项。
- setOnMenuItemClickListener(MenuItem, OnMenuItemClickListener menuItemClickListener)：设置子菜单项的监听器。

创建一个子菜单的步骤为：

① 覆盖 Activity 的 onCreateOptionsMenu()方法，调用 Menu 的 addSubMenu()方法来添加子菜单；

② 调用 SubMenu 的 add()方法，添加子菜单项；

③ 覆盖 onContextItemSelected()方法，响应子菜单的单击事件。

子菜单提供了一种自然的组织菜单项的方式，可以通过 addSubMenu(int groupId，int itemId，int order，int titleRes)方法非常方便地创建和响应子菜单。

下面通过一个实例来演示子菜单的用法。

【例 4-7】 一个子菜单用于实现蓝牙。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 SubMenu_test。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中声明一个 TextView 控件。代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity" >
    <TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_centerHorizontal = "true"
        android:layout_centerVertical = "true"
        android:text = "@string/hello_world" />
</RelativeLayout>
```

(3) 打开 src\submenu_test 包下的 MainActivity.java 文件，在文件中实现子菜单，当单击子菜单项时，即弹出对应的提示信息。代码为：

```
package fs.submenu_test;
import android.os.Bundle;
import android.app.Activity;
```

```java
import android.widget.Toast;
import android.view.Menu;
import android.view.MenuItem;
import android.view.SubMenu;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    //初始化菜单
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
      //添加 OptionsMenu 菜单项
    /* menu.add(groupId,itemId,order,title)
     * groupId:菜单项所在的组
     * itemId:菜单项编号
     * order:排序
     * title:标题
     * setIcon()方法为菜单设置图标,这里使用的是系统自带的图标,请留意一下,
     * 以 android.R 开头的资源是系统提供的,自己提供的资源是以 R 开头的  */
        menu.add(0,1,Menu.NONE,"蓝牙发送").setIcon(android.R.drawable.ic_menu_send);
        //添加子菜单
        SubMenu subMenu = menu.addSubMenu(0,2,Menu.NONE,"重要程度>>").setIcon(android.R.
drawable.ic_menu_share);
          //添加子菜单项
          subMenu.add(2,201,1,"☆☆☆☆☆");
          subMenu.add(2,202,2,"☆☆☆");
          subMenu.add(2,203,3,"☆");
          menu.add(0,3,Menu.NONE,"重命名").setIcon(android.R.drawable.ic_menu_edit);
          menu.add(0,4,Menu.NONE,"删除").setIcon(android.R.drawable.ic_menu_close_clear_
cancel);
        return true;
    }
    //根据菜单执行相应内容
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
        case 1:
            Toast.makeText(getApplicationContext(),"蓝牙发送……",Toast.LENGTH_SHORT).show();
            return true;
        case 201:
            Toast.makeText(getApplicationContext(),"非常重要：☆☆☆☆☆",Toast.LENGTH_
SHORT).show();
            return true;
        case 202:
Toast.makeText(getApplicationContext(),"重要：☆☆☆",Toast.LENGTH_SHORT).show();
            return true;
        case 203:
            Toast.makeText(getApplicationContext(),"普通：☆",Toast.LENGTH_SHORT).show();
            return true;
        case 3:
            Toast.makeText(getApplicationContext(),"重命名……",Toast.LENGTH_SHORT).show();
            return true;
```

```
            case 4:
                Toast.makeText(getApplicationContext(),"删除……",Toast.LENGTH_SHORT).show();
                return true;
            }
            return false;
        }
    }
```

运行程序，默认效果如图 4-9(a)所示，单击界面右上角的⋮按钮，即弹出设置的子菜单，效果如图 4-9(b)所示，单击子菜单项，即弹出对应的提示信息，效果如图 4-9(c)及图 4-9(d)所示。



(a) 默认界面

(b) 设置子菜单项



(c) 子菜单提示消息1



(d) 子菜单提示消息2

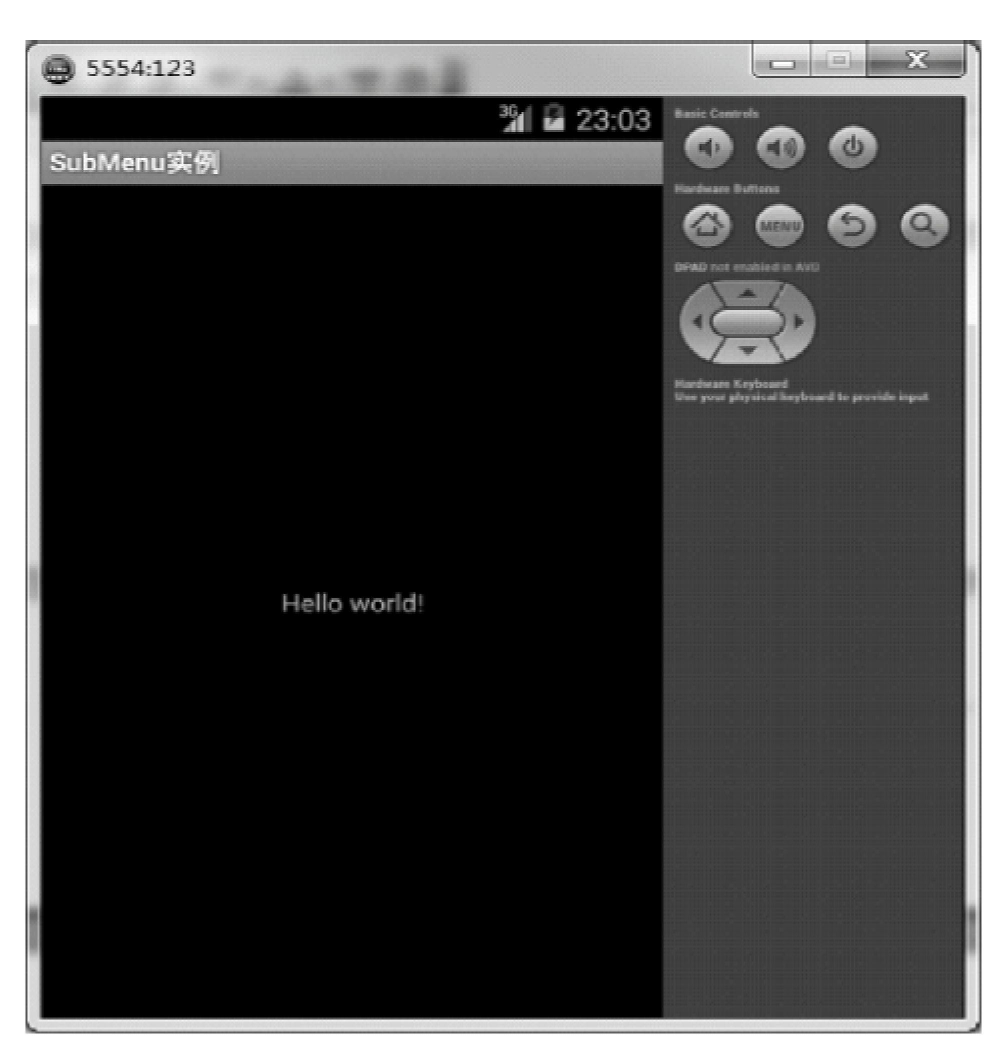
图 4-9　子菜单项 1

当需要将界面设置成“老”样式界面时，需要改动的并不大，仅需要修改 AndroidManifest.xml 文件的样式即可。

```
...
    <activity
```

```
            android:name = "fs.submenu_test.MainActivity"
                android:label = "@string/app_name"
                android:theme = "@android:style/Theme" >
                < intent - filter >
                    < action android:name = "android.intent.action.MAIN" />
                    < category android:name = "android.intent.category.LAUNCHER" />
                </intent - filter >
            </activity >
        </application >
    </manifest >
```

重新运行程序，默认效果如图 4-10(a)所示，当单击界面中右侧的“MENU”选项时，效果如图 4-10(b)所示，当选择对应的菜单项时，效果如图 4-10(c)所示。

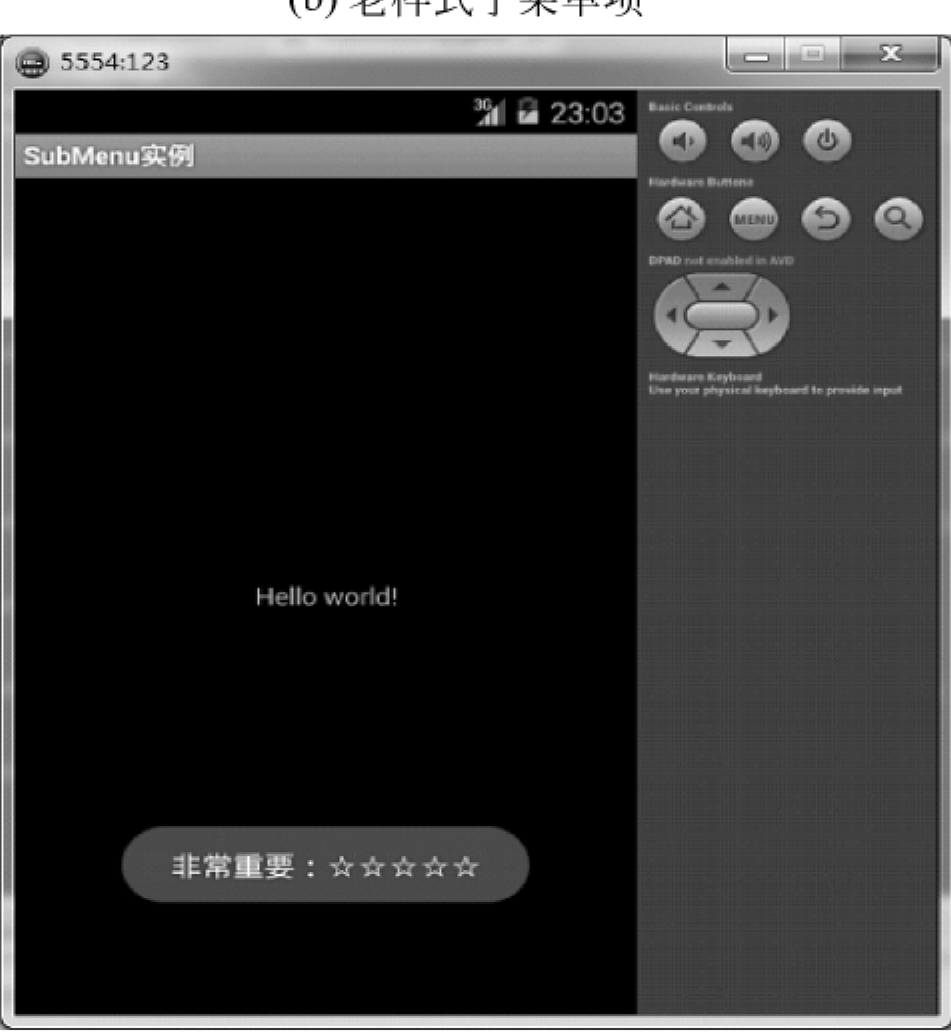


(a) 老样式默认界面

(b) 老样式子菜单项



(c) 子菜单提示消息1



(d) 子菜单提示消息2

图 4-10　子菜单项 2

4.4.3 Android 上下文菜单

上下文菜单 ContextMenu 可以像操作 Options Menu 那样给上下文菜单增加菜单项。

上下文菜单与 Options Menu 最大的不同在于，Options Menu 的拥有者是 Activity，而上下文菜单的拥有者是 Activity 中的 View。每个 Activity 有且只有一个 Options Menu，它为整个 Activity 服务。而一个 Activity 往往有多个 View，并不是每个 View 都有上下文菜单，这就需要我们显示地通过 registerForContextMenu(View view)来指定。

尽管上下文菜单的拥有者是 View，生成上下文菜单却是通过 Activity 中的 onCreateContextMenu(ContextMenu menu, View v, ContextMenu. ContextMenuInfo menuInfo)方法，该方法很像生成 Options Menu 的 onCreateOptionsMenu(Menu menu)方法。两者的不同在于，onCreateOptionsMenu 只在用户第 1 次按 Menu 键时被调用，而 onCreateContextMenu 会在用户每一次长按 View 时被调用，而且 View 必须已经注册了上下文菜单。使用 context menu 的主要方法有：

- registerForContextMenu(View view)：为某个 View 注册菜单。
- onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo menuInfo)：创建 ContextMenu，会在 menu 第 1 次显示时调用。
- onContextItemSelected(MenuItem item)：菜单项被选中后处理选中的菜单项。
- onContextMenuClosed(Menu menu)：菜单被关闭的事件。
- openContextMenu(View view)：调用打开菜单。
- closeContextMenu()：调用关闭菜单。

下面通过一个实例来演示上下文菜单的使用。上两个小节中的 OptionsMenu 及 SubMenu 都通过 xml 文件配置菜单项，本实例直接采用代码完成，故不使用布局文件。

【例 4-8】 设置一个上下文菜单。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 ContextMenu_test。

(2) 打开 src\fs. contextmenu_test 包下的 MainActivity. java 文件。代码为：

```
package fs.contextmenu_test;
import android.os.Bundle;
import android.view.View;
import android.app.ListActivity;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuInfo;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.ArrayAdapter;
import android.widget.Toast;
public class MainActivity extends ListActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //显示列表
        simpleShowList();
        //为所有列表项注册上下文菜单
```

```
        this.registerForContextMenu(getListView());
    }
    private void simpleShowList() {
        //模拟文件列表,在项目中可以实际读取文件列表
        String[] files = new String[] {
                "网站备案通知.doc",
                "企业预决算报表.xls",
                "客户说明会.ppt",
                "企业形象宣传片.avi"
        };
        //数据适配器 simple array adapter
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.
simple_list_item_1,files);
        //填充列表
        this.setListAdapter(adapter);
        }
    @Override
    public void onCreateContextMenu(ContextMenu menu,View v,ContextMenuInfo menuInfo) {
        //设置 ContextMenu 标题
        menu.setHeaderTitle("文件操作");
        //添加 ContextMenu 菜单项
        menu.add(0,1,Menu.NONE,"蓝牙发送");
        menu.add(0,2,Menu.NONE,"标记为重要");
        menu.add(0,3,Menu.NONE,"重命名");
        menu.add(0,4,Menu.NONE,"删除");
    }
    @Override
    public boolean onContextItemSelected(MenuItem item) {
        //得到当前被选中的 item 信息
        switch(item.getItemId()) {
            case 1:
Toast.makeText(getApplicationContext(),"发送文件",Toast.LENGTH_SHORT).show();
                break;
            case 2:
                Toast.makeText(getApplicationContext(),"标记为重要☆☆☆",Toast.LENGTH_
SHORT).show();
                break;
            case 3:
    Toast.makeText(getApplicationContext(),"重命名",Toast.LENGTH_SHORT).show();
                break;
            case 4:
      Toast.makeText(getApplicationContext(),"删除",Toast.LENGTH_SHORT).show();
                break;
            default:
                return super.onContextItemSelected(item);
            }
        return true;
    }
}
```

运行程序，默认效果如图 4-11(a)所示，长按界面中的任一个文本框，将弹出如图 4-11(b)所示的上下文菜单。



(a) 默认界面



(b) 上下文菜单

图 4-11　上下文菜单设置

4.4.4　Android 菜单综合实例

下面通过一个综合实例来演示选项菜单与子菜单的用法。

【例 4-9】　主要功能是接收用户在菜单中的选项并输出到文本框中。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Sub_Options_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明一个 ScrollView 控件及一个 EditText 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@ + id/LinearLayout01"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "#aabbcc">
<! -- 声明一个线性布局 -->
    < ScrollView
        android:id = "@ + id/ScrollView01"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent" >
<! -- 声明 ScrollView 控件 -->
        < EditText
            android:id = "@ + id/EditText01"
            android:layout_width = "fill_parent"
            android:layout_height = "fill_parent"
            android:cursorVisible = "false"
```

```
                android:editable = "false"
                android:text = "@string/label" >
<! -- 声明一个 EditText 控件 -->
            </EditText>
      </ScrollView>
</LinearLayout>
```

(3) 打开 res\values 目录下的 strings.xml 文件，为变量声明值。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
      <string name = "app_name">Sub_Options 实例</string>
      <string name = "action_settings">Settings</string>
      <string name = "hello_world">Hello world!</string>
      <string name = "label">您的选择为\n</string>
      <! -- 声明名为 label 的字符串资源 -->
      <string name = "gender">性别</string>
      <! -- 声明名为 gender 的字符串资源 -->
      <string name = "male">男</string>
      <! -- 声明名为 male 的字符串资源 -->
      <string name = "female">女</string>
      <! -- 声明名为 female 的字符串资源 -->
      <string name = "hobby">爱好</string>
      <! -- 声明名为 hobby 的字符串资源 -->
      <string name = "hobby1">唱歌</string>
      <! -- 声明名为 hobby1 的字符串资源 -->
      <string name = "hobby2">跳舞</string>
      <! -- 声明名为 hobby2 的字符串资源 -->
      <string name = "hobby3">网虫</string>
      <! -- 声明名为 hobby3 的字符串资源 -->
      <string name = "ok">确定</string>
      <! -- 声明名为 ok 的字符串资源 -->
</resources>
```

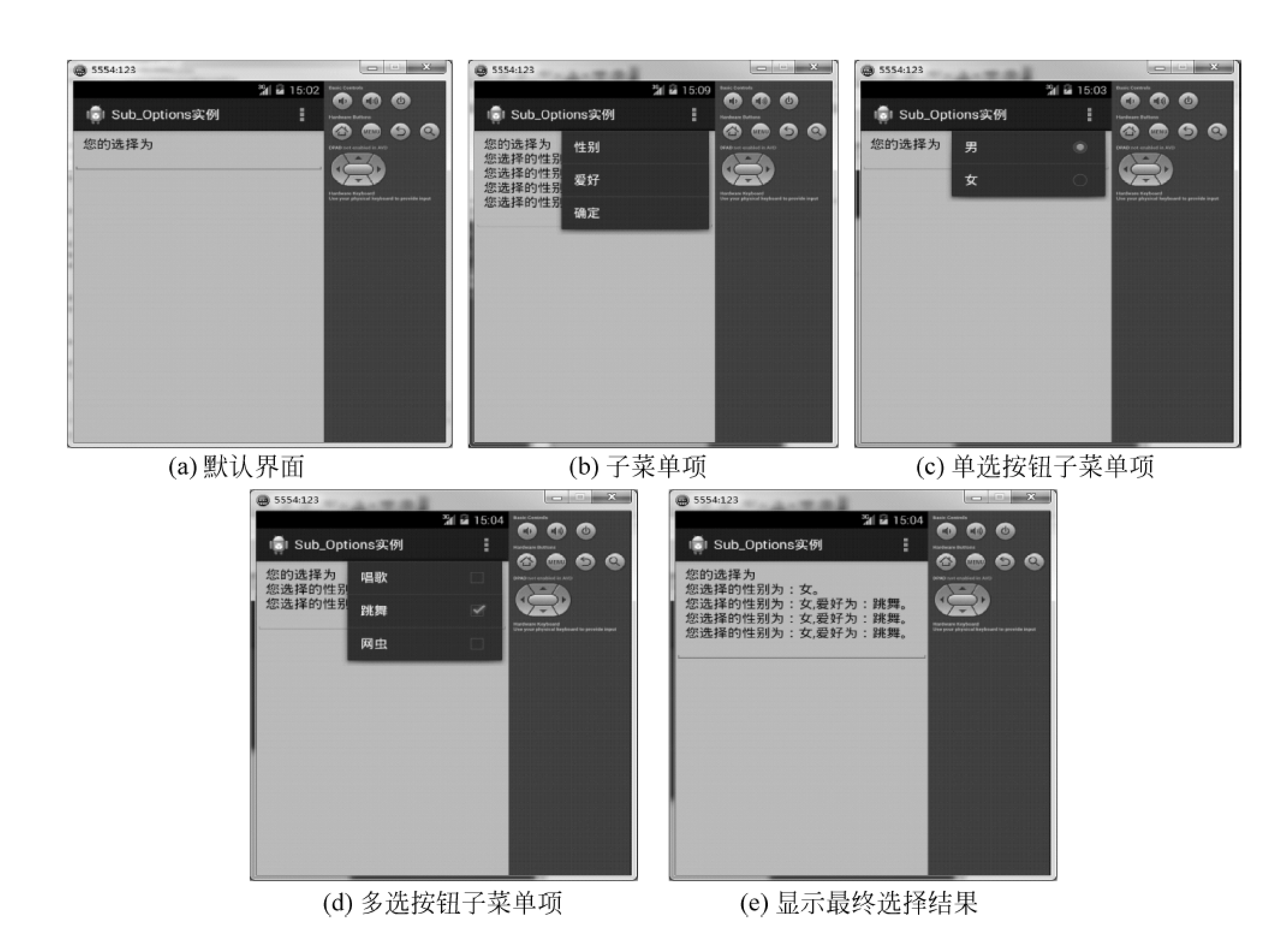
(4) 打开 src\fs.sub_options_test 包下的 MainActivity.java 文件，在文件中实现菜单选项及子菜单。代码为：

```
package fs.sub_options_test;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MenuItem.OnMenuItemClickListener;
import android.view.SubMenu;
import android.widget.EditText;
public class MainActivity extends Activity {
      final int MENU_GENDER_MALE = 0;                //性别为男选项编号
      final int MENU_GENDER_FEMALE = 1;              //性别为女选项编号
      final int MENU_HOBBY1 = 2;                     //爱好 1 选项编号
      final int MENU_HOBBY2 = 3;                     //爱好 2 选项编号
      final int MENU_HOBBY3 = 4;                     //爱好 3 选项编号
```

```
final int MENU_OK = 5;                              //确定菜单选项编号
final int MENU_GENDER = 6;                          //性别子菜单编号
final int MENU_HOBBY = 7;                           //爱好子菜单编号每个菜单项目的编号 =
final int GENDER_GROUP = 0;                         //性别子菜单项组的编号
final int HOBBY_GROUP = 1;                          //爱好子菜单项组的编号
final int MAIN_GROUP = 2;                           //外层总菜单项组的编号
MenuItem[ ] miaHobby = new MenuItem[3];             //爱好菜单项组
MenuItem male = null;                               //男性性别菜单项
@Override
public void onCreate(Bundle savedInstanceState) {  //重写 onCreate 方法
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);                  //设置当前屏幕
}
@Override
//通过 onCreateOptionsMenu 实现初始化菜单,该处包括 3 个菜单项,分别为性别子菜单、爱好子
//菜单及确定子菜单
public boolean onCreateOptionsMenu(Menu menu){
    //性别单选菜单项组,菜单若编组就是单选菜单项组
    SubMenu subMenuGender  =  menu.addSubMenu(MAIN_GROUP,MENU_GENDER,0,R.string.gender);
    subMenuGender.setIcon(R.drawable.g4);           //图片资源
    subMenuGender.setHeaderIcon(R.drawable.g4);
    male = subMenuGender.add(GENDER_GROUP,MENU_GENDER_MALE,0,R.string.male);
    male.setChecked(true);
    subMenuGender.add(GENDER_GROUP,MENU_GENDER_FEMALE,0,R.string.female);
    //设置 GENDER_GROUP 组是可选择的、互斥的
    subMenuGender.setGroupCheckable(GENDER_GROUP,true,true);
    //爱好复选菜单项组
    SubMenu subMenuHobby  =  menu.addSubMenu(MAIN_GROUP,MENU_HOBBY,0,R.string.hobby);
    miaHobby[0] = subMenuHobby.add(HOBBY_GROUP,MENU_HOBBY1,0,R.string.hobby1);
    miaHobby[1] = subMenuHobby.add(HOBBY_GROUP,MENU_HOBBY2,0,R.string.hobby2);
    miaHobby[2] = subMenuHobby.add(HOBBY_GROUP,MENU_HOBBY3,0,R.string.hobby3);
    miaHobby[0].setCheckable(true);                 //设置菜单项为复选菜单项
    miaHobby[1].setCheckable(true);
    miaHobby[2].setCheckable(true);
    //确定菜单项
    MenuItem ok = menu.add(GENDER_GROUP + 2,MENU_OK,0,R.string.ok);
    OnMenuItemClickListener lsn = new OnMenuItemClickListener(){
                                                        //实现菜单项单击事件监听接口功能
        public boolean onMenuItemClick(MenuItem item) {
            appendStateStr();
            return true;
        }
    };
    ok.setOnMenuItemClickListener(lsn);             //给确定菜单项添加监听器
    //给确定菜单项添加快捷键
    ok.setAlphabeticShortcut('o');                  //设置字符快捷键
    //要注意,同时设置多次时只有最后一个设置起作用
    return true;
}
@Override                           //单选或复选菜单项选中状态变化后的回调方法
//其他选项菜单
```

```
    public boolean onOptionsItemSelected(MenuItem mi){
        switch(mi.getItemId()){
          case MENU_GENDER_MALE:    //单选菜单项状态的切换要自行写代码完成
          case MENU_GENDER_FEMALE:
                mi.setChecked(true);
                appendStateStr(); //当有效项目变化时记录在文本区中
                break;
          case MENU_HOBBY1:         //复选菜单项状态的切换要自行写代码完成
          case MENU_HOBBY2:
          case MENU_HOBBY3:
              mi.setChecked(!mi.isChecked());
              appendStateStr();    //当有效项目变化时记录在文本区中
              break;
        }
        return true;
    }
    //获取当前选择状态的方法
    public void appendStateStr(){
        String result = "您选择的性别为:";
        if(male.isChecked()){
            result = result + "男";
        }
        else{
            result = result + "女";
        }
        String hobbyStr = "";
        for(MenuItem mi:miaHobby){
            if(mi.isChecked()){
                hobbyStr = hobbyStr + mi.getTitle() + "、";
            }
        }
        if(hobbyStr.length()> 0){
            result = result + ",爱好为:" + hobbyStr.substring(0,hobbyStr.length() - 1) +
".\n";
        }
        else{
            result = result + ".\n";
        }
        EditText et = (EditText)MainActivity.this.findViewById(R.id.EditText01);
        et.append(result);
    }
}
```

运行程序,默认效果如图 4-12(a)所示,单击界面右上角的按钮,即弹出如图 4-12(b)所示的子菜单项,单击子菜单中的"性别"选项,效果如图 4-12(c)所示带单选按钮的子菜单,单击子菜单中的"爱好"选项,效果如图 4-12(d)所示带多选按钮的子菜单,得到最终结果如图 4-12(e)所示。



(a) 默认界面　(b) 子菜单项　(c) 单选按钮子菜单项

(d) 多选按钮子菜单项　(e) 显示最终选择结果

图 4-12　菜单项与子菜单效果

第 5 章 Android 视图

在前面章节中，介绍了 Android 平台下在开发用户界面时常用的控件与菜单及对话框，除了这些外，为了界面的美观和友好性还会用到其他视图及动画效果，本章主要介绍 Android 的视图。

5.1 Android 图像视图

图像视图(ImageView)，用于在屏幕中显示任何 Drawable 对象，通常用来显示图片。在 Android 中，可以使用两种方法向屏幕中添加图像视图：一种是通过 XML 布局文件中使用<ImageView>标记添加；另一种是在 Java 文件中，通过 new 关键字创建。建议使用第一种方法。

在使用 ImageView 组件显示图像时，通常可以将要显示的图片放置在 res/drawable 目录中，然后应用以下代码在布局管理器中。

```
< ImageView
    android:id = "@ + id/imageView1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_alignLeft = "@ + id/textView1"
    android:layout_below = "@ + id/textView1"
    android:layout_marginLeft = "33dp"
    android:layout_marginTop = "117dp"
    android:src = "@drawable/ic_launcher" />
```

ImageView 支持的常用 XML 属性主要如表 5-1 所示。

表 5-1 ImageView 支持的 XML 属性

XML 属性	描　　述
android:adjustViewBounds	用于设置 ImageView 是否调整自己的边界来保持所显示图片的长宽比
android:maxHeight	设置 ImageView 的最大高度，需要设置 android:adjustViewBounds 属性值为 true，否则不起使用
android:maxWidth	设置 ImageView 的最大宽度，需要设置 android:adjustViewBounds 属性值为 true，否则不起作用

续表

XML 属性	描　　述
android:scaleType	用于设置所显示的图片怎样缩放或移动以适应 ImageView 的大小，其属性值可以是 maxtrix(使用 matrix 方式进行缩放)、fitXY(对图片横向、纵向独立缩放，使得该图片完全适应于该 ImageView，图片的纵横比可能会改变)、fitStart(保持纵横比缩放图片，直到该图片能完全显示在 ImageView 中，缩放完全显示在 ImageView 的左上角)、fitCenter(保持纵横比缩放图片，直到该图片能完全显示在 ImageView 中，缩放完成后该图片放在 ImageView 的中央)、fitEnd(保持纵横比缩放图片，直到该图片能完全显示在 ImageView 中，缩放完成后该图片放在 ImageView 的右下角)、center(把图片放在 ImageView 的中间，但不进行任何缩放)、centerCrop(保持纵横比缩放图片，以使得图片能完全覆盖 ImageView)或 centerInside(保持纵横比缩放图片，以使得 ImageView 能完全显示该图片)
android:src	用于设置 ImageView 所显示的 Drawable 对象的 ID，例如，设置显示保存在 res/drawable 目录下的名称为 g4.jpg 的图片，可以将属性值设置为 android:src="@drawable/g4"
android:tint	用于为图片着色，其属性值可以是#rgb、#argb、#rrggbb 或#aarrggbb 表示的颜色值

下面给出两个关于 ImageView 控件的实例，演示 ImageView 的用法。

【例 5-1】 利用 ImageView 显示图像。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 ImageView_test。

(2) 打开 res\layout 目录下的 main.xml 文件，在布局文件中声明 3 个 ImageView 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
    <! -- ImageView 组件用于按图片的原始尺寸显示图像 -->
    < ImageView
        android:id = "@ + id/imageView2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_margin = "5px"
        android:adjustViewBounds = "true"
        android:maxHeight = "180px"
        android:maxWidth = "180px"
        android:src = "@drawable/a01" />
<! -- ImageView 组件用于设置组件的最大高度和宽度 -->
    < ImageView
        android:src = "@drawable/a01"
        android:id = "@ + id/imageView1"
        android:layout_margin = "5px"
        android:layout_height = "wrap_content"
```

```
        android:layout_width = "wrap_content"/>
      <! -- ImageView 组件用于保持纵横比缩放图片,直到该图片能完全显示在 ImageView 组件
中,并让图片显示在 ImageView 组件的右下角 -->
    < ImageView
        android:src = "@drawable/a01"
        android:id = "@ + id/imageView3"
        android:scaleType = "fitEnd"
        android:layout_margin = "5px"
        android:layout_height = "180px"
        android:layout_width = "180px"/>
      <! -- ImageView 组件用于实现在该组件中的着色功能,在此设置的是半透明的红色 -->

    < ImageView
        android:src = "@drawable/a01"
        android:id = "@ + id/imageView4"
        android:tint = "#77ff0000"
        android:layout_height = "180px"
        android:layout_width = "180px"/>
</LinearLayout >
```

运行程序,效果如图 5-1 所示。



图 5-1　应用 ImageView 显示图像

以上例子是通过 XML 文件属性来改变图像的,下面通过在 Java 文件中,改变 ImageView 的属性。

【例 5-2】 实现单击图像改变图像的透明度。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 ImageView_test2。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明一个 ImageView 控件。代码为：

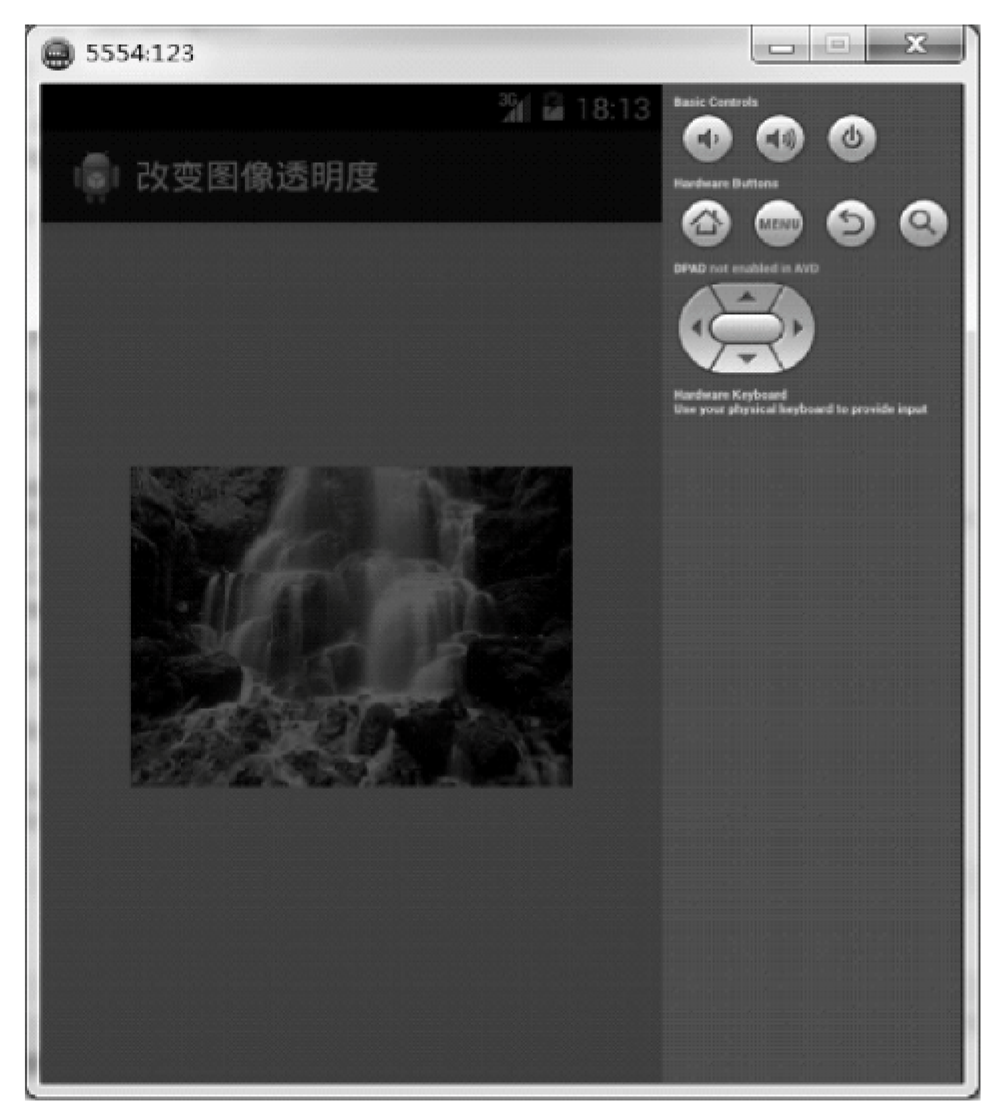
```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc">
    <ImageView
    android:id = "@ + id/ImageView1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:paddingLeft = "30dip"
    android:paddingRight = "30dip"
    android:visibility = "visible"/>
</RelativeLayout>
```

(3) 打开 src\fs.imageview_test2 包下的 MainActivity.java 文件，在文件中单击图像改变图像的透明度。代码为：

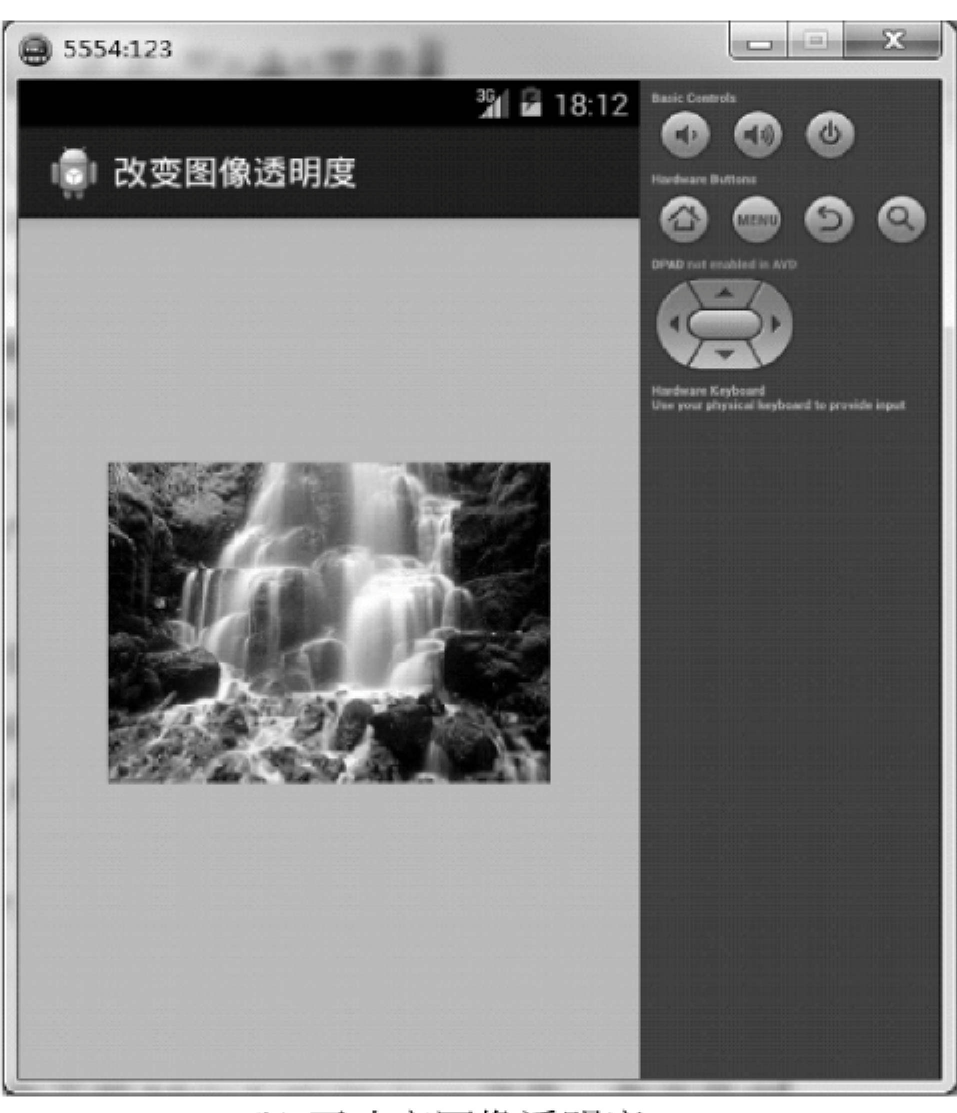
```
package fs.imageview_test2;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ImageView;
public class MainActivity extends Activity
{
    ImageView iv;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //初始化 ImageView
        iv = (ImageView)findViewById(R.id.ImageView1);
        iv.setImageDrawable(getResources().getDrawable(R.drawable.a02));
                                                        //为 ImageView 设置图片
        iv.setAlpha(100);            //设置不透明度为 100
        //为 ImageView 设置监听,当单击图片的时候,图片不透明度增加
        iv.setOnClickListener
        (
                new OnClickListener()
                {
                    public void onClick(View v)
                    {
```

```
iv.setImageDrawable(getResources().getDrawable(R.drawable.a02));
                        iv.setAlpha(255);  //设置不透明度为 255
                    }
                }
        );
    }
}
```

运行程序，默认效果如图 5-2(a)所示，当单击图像时，即改变图像的透明度，效果如图 5-2(b)所示。



(a) 改变图像透明度



(b) 不改变图像透明度

图 5-2　实现图像透明度改变

5.2　Android 网格视图

网格视图按照行、列分布的方式来显示多个组件，通常用于显示图片或图标等。在使用网格视图时，首先需要在屏幕上添加 GridView 组件，通常使用<GridView>标记在 XML 布局文件中添加。在 XML 布局文件中添加网格视图的基本格式为：

```
<GridView
    android:id = "@ + id/gridView1"
    android:layout_width = "match_parent"
    android:layout_height = "wrap_content"
    android:layout_below = "@ + id/textView1"
    android:layout_centerHorizontal = "true"
    android:layout_marginTop = "117dp"
    android:numColumns = "3" >
```

GridView 组件支持的 XML 属性如表 5-2 所示。

表 5-2　GridView 支持的 XML 属性

XML 属性	描　　述
android:columnWidth	用于设置列的宽度
android:gravity	用于设置对齐方式
android:horizontalSpacing	用于设置各元素之间的水平间距
android:numColumns	用于设置列数，其属性值通常为大于 0 的值，如果只有一列，那么最好使用 ListView 实现
android:stretchMode	用于设置拉伸模式，其中属性值可以是 none（不拉伸）、spacingWidth（仅拉伸元素之间的间距）、columnWidth（仅拉伸表格元素本身）或 spacingWidthUniform（表格元素本身、元素之间的间距一起拉伸）
android:verticalSpacing	用于设置各元素之间的垂直间距

GridView 与 ListView 类似，都需要通过 Adapter 来提供要显示的数据。在使用 GridView 组件时，通常使用 SimpleAdapter 或者 BaseAdapter 类为 GridView 组件提供数据。

注意：使用 GridView 时一般都应该指定 numColumns 大于 1，否则该属性的默认值为 1。如果将属性设为 1，则意味着该 GridView 只有一列，那么 GridView 就变成了 ListView。

下面通过两个例子来说明 GridView 控件的用法。

【例 5-3】 实现在屏幕中添加用于显示照片和说明文字的网格视图。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 GridView。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明一个 GridView 控件及两个 TextView 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
        android:orientation = "vertical"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:background = "#aabbcc">
< TextView
    android:text = "这次决斗谁赢了!"
    android:id = "@ + id/textView1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"/>
< GridView
    android:layout_height = "wrap_content"
    android:id = "@ + id/gridView"
    android:layout_width = "match_parent"
    android:numColumns = "2"
    android:background = "#FFF"/>
< TextView
      android:layout_height = "wrap_content"
      android:layout_width = "fill_parent"
      android:text = "@string/hello_world"
      android:id = "@ + id/text"/>
</LinearLayout >
```

(3) 在 res\layout 目录下创建一个 item.xml 文件,用于为每个 grid 的单项设置一个样式。代码为:

```
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
        android:orientation = "vertical"
        android:layout_width = "match_parent"
        android:layout_height = "match_parent"
        android:background = "#aabbcc">
<ImageView
    android:layout_height = "wrap_content"
    android:id = "@ + id/item_imageView"
    android:layout_width = "wrap_content"
    android:src = "@drawable/po1"
    android:layout_gravity = "center"/>
<TextView
    android:text = "TextView"
    android:id = "@ + id/item_textView"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_gravity = "center"/>
</LinearLayout>
```

(4) 打开 src\fs.gridview 包下的 MainActivity.java 文件,在文件中实现当单击相关图片时,即把相关信息显示在 TextView 控件中。代码为:

```
package fs.gridview;
import java.util.ArrayList;
import java.util.HashMap;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.GridView;
import android.widget.SimpleAdapter;
import android.widget.TextView;
public class MainActivity extends Activity
{
    private TextView text  =  null;
    private int[] image  =  { R.drawable.po1,R.drawable.po2,
            R.drawable.po3,R.drawable.po5 };
    private String[] item  =  { "杀生丸","宇智波佐助 ","江户川柯南","樱木花道" };
    private GridView gridView;
    private SimpleAdapter adapter;
    /** 第 1 次调用 Activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //通过 ID 查找到 main.xml 中的 TextView 控件
```

```
            text = (TextView) findViewById(R.id.text);
            //通过 ID 查找到 main.xml 中的 GridView 控件
            gridView = (GridView) findViewById(R.id.gridView);
            //创建一个 ArrayList 列表,内部存储的是 HashMap 列表
            ArrayList < HashMap < String, Object >> listItems = new ArrayList < HashMap < String,
Object >>();
            //将数组信息分别存储到 ArrayList 中
            int len = item.length;
            for(int i = 0; i < len ; i++){
                HashMap< String,Object > map = new HashMap< String,Object >();
                map.put("image",image[i]);
                map.put("item",item[i]);
                listItems.add(map);
            }
            //HashMap 中的 Key 信息,要与 grid_item.xml 中的信息做对应
            String[] from = {"image","item"};
            //grid_item.xml 中对应的 ImageView 控件和 TextView 控件
            int[] to = {R.id.item_imageView,R.id.item_textView};
            //设定一个适配器
            adapter = new SimpleAdapter(this,listItems,R.layout.item,from,to);
            //对 GridView 进行适配
            gridView.setAdapter(adapter);
            //设置 GridView 的监听器
            gridView.setOnItemClickListener(new OnItemClickListener()
            {
                @Override
                public void onItemClick(AdapterView <?> arg0,View arg1,
                        int position,long arg3)
                {
                    String str = "这次决斗" + item[position] + "赢了!";
                    updateText(str);
                }
            });
        }
        private void updateText(String string)
        {
            //将文本信息设置给 TextView 控件显示出来
            text.setText(string);
        }
    }
```

运行程序,效果如图 5-3 所示。

下面通过 GridView 控件来实现一个宫式布局图。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 GridView_test。

(2) 打开 res\layout 目录下的 main. xml 布局文件,在文件中声明一个 GridView 控件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! --
android:numColumns = "auto_fit"                    //GridView 的列数设置为自动
```



图 5-3 GridView 视图

```
android:columnWidth = "90dp"                   //每列的宽度,也就是 Item 的宽度
android:stretchMode = "columnWidth"            //缩放与列宽大小同步
android:verticalSpacing = "10dp"               //两行之间的边距
android:horizontalSpacing = "10dp"             //两列之间的边距
-->
<GridView
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@ + id/gridview"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:numColumns = "auto_fit"
    android:verticalSpacing = "10dp"
    android:horizontalSpacing = "10dp"
    android:columnWidth = "90dp"
    android:stretchMode = "columnWidth"
    android:gravity = "center"
    android:background = "#aabbcc"/>
```

（3）在 res\layout 目录下新建一个 items.xml 文件，在文件中声明一个 ImageView 控件及一个 TextView 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<RelativeLayout
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_height = "wrap_content"
    android:paddingBottom = "4dip"
    android:layout_width = "fill_parent"
```

```
        android:background = "#aabbcc">
        <ImageView
            android:layout_height = "wrap_content"
            android:layout_width = "wrap_content"
            android:layout_centerHorizontal = "true"
            android:id = "@ + id/itemImage"/>
        <TextView
            android:layout_width = "wrap_content"
            android:layout_below = "@ + id/itemImage"
            android:layout_height = "wrap_content"
            android:text = "TextView01"
            android:layout_centerHorizontal = "true"
            android:id = "@ + id/itemText"/>
</RelativeLayout>
```

(4) 打开 src\fs.gridview_test 包下的 MainActivity.java 文件，在文件中实现九宫式布局图，并当单击某个宫式时，即弹出对应的 Toast 提示信息。代码为：

```
package fs.gridview_test;
import java.util.ArrayList;
import java.util.HashMap;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.GridView;
import android.widget.SimpleAdapter;
import android.widget.Toast;
import android.widget.AdapterView.OnItemClickListener;
public class MainActivity extends Activity {
    private String texts[] = null;
    private int images[] = null;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        images = new int[]{R.drawable.d1,R.drawable.d2,
                R.drawable.d3,R.drawable.d4,
                R.drawable.d5,R.drawable.d6,
                R.drawable.d7,R.drawable.d8};
        texts = new String[]{ "宫式布局 1","宫式布局 2","宫式布局 3","宫式布局 4",
                "宫式布局 5","宫式布局 6","宫式布局 7","宫式布局 8"};
        GridView gridview = (GridView) findViewById(R.id.gridview);
        ArrayList<HashMap<String,Object>> lstImageItem = new ArrayList<HashMap<String,
Object>>();
        for (int i = 0; i < 8; i++) {
            HashMap<String,Object> map = new HashMap<String,Object>();
            map.put("itemImage",images[i]);
            map.put("itemText",texts[i]);
            lstImageItem.add(map);
        }
```

```
            SimpleAdapter saImageItems = new SimpleAdapter(this,
                    lstImageItem,                   //数据源
                    R.layout.items,                 //显示布局
                    new String[] { "itemImage","itemText" },
                    new int[] { R.id.itemImage,R.id.itemText });
            gridview.setAdapter(saImageItems);
            gridview.setOnItemClickListener(new ItemClickListener());
        }
        class ItemClickListener implements OnItemClickListener {
            /**
              * 单击选项时触发事件
              * @param parent  发生单击动作的 AdapterView
        * @param view 在 AdapterView 中被单击的视图(它是由 adapter 提供的一个视图)
              * @param position 视图在 adapter 中的位置
              * @param rowid 被点单元素的行 id
              */
            public void onItemClick(AdapterView<?> parent,View view,int position,long rowid) {
                HashMap<String,Object> item = (HashMap<String,Object>) parent.getItemAtPosition
    (position);
                //获取数据源的属性值
                String itemText=(String)item.get("itemText");
                Object object=item.get("itemImage");
                Toast.makeText(MainActivity.this,itemText,Toast.LENGTH_LONG).show();
                //根据图片进行相应的跳转
                switch (images[position]) {
                case R.drawable.d1:
        startActivity(new Intent(MainActivity.this,TestActivity1.class));//启动另一个 Activity
                    finish();                       //结束此 Activity,可回收
                    break;
                case R.drawable.d2:
                    startActivity(new Intent(MainActivity.this,TestActivity2.class));
                    finish();
                    break;
                case R.drawable.d3:
                    startActivity(new Intent(MainActivity.this,TestActivity3.class));
                    finish();
                    break;
                }

            }
        }
    }
```

(5) 在 src\fs.gridview_test 包下创建 3 个活动 Activity,用于实现宫式的跳转,分别命名为 TestActivity1、TestActivity2 及 TestActivity3。代码分别为:

```
    package fs.gridview_test;
    import android.app.Activity;
    import android.os.Bundle;
    public class TestActivity1 extends Activity {
        @Override
```

```
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
        }
}

package fs.gridview_test;
import android.app.Activity;
import android.os.Bundle;
public class TestActivity2 extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}

package fs.gridview_test;
import android.app.Activity;
import android.os.Bundle;
public class TestActivity3 extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

（6）打开 AndroidManifest.xml 文件，在文件中设置权限。代码为：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="fs.gridview_test"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="fs.gridview_test.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
```

```
        <!-- 设置权限 -->
          <activity android:name=".TestActivity1" android:label="@string/test_name1"/>
        <activity android:name=".TestActivity2" android:label="@string/test_name2"/>
        <activity android:name=".TestActivity3" android:label="@string/test_name3"/>
    </application>
</manifest>
```

运行程序,效果如图 5-4 所示。



图 5-4　九宫式布局图

5.3　Android 可扩展列表组件

可展开的列表组件(ExpandableListView)为 ListView 的子类,它在普通 ListView 的基础上进行扩展,它把应用中的列表项分为几组,每组里又可包含多个列表项。

ExpandableListView 的用法与普通 ListView 的用法非常类似,只是 ExpandableListView 所显示的列表项应该由 ExpandableListView 提供。表 5-3 列出了 ExpandableListView 所额外支持的常用 XML 属性。

表 5-3　ExpandableListView 额外支持的常用 XML 属性

XML 属性	描　述
android:childDivider	指定各组内子列表项之间的分隔条
android:childIndicator	显示在子列表项旁边的 Drawable 对象
android:groupIndicator	显示在组列表项旁边的 Drawable 对象

下面通过一个案例来演示 ExpandableListView 控件的用法。

【例 5-4】 利用 ExpandableListView 控件显示图片及相关信息。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 ExpandableListView_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明一个 ExpandableListView 控件及两个 TextView 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
< TextView
    android:text = "动漫中各名人的能力"
    android:id = "@ + id/textView1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"/>
< ExpandableListView
    android:layout_height = "wrap_content"
    android:id = "@ + id/expandableListView"
    android:layout_width = "match_parent"/>
< TextView
    android:layout_height = "wrap_content"
    android:layout_width = "fill_parent"
    android:text = "@string/hello_world"
    android:id = "@ + id/text"/>
</LinearLayout >
```

(3) 打开 src\fs. expandablelistview_test 包下的 MainActivity.java 文件，在文件中实现当单击图片时，即显示图片中的相关信息。代码为：

```
package fs.expandablelistview_test;
import android.app.Activity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AbsListView;
import android.widget.BaseExpandableListAdapter;
import android.widget.ExpandableListAdapter;
import android.widget.ExpandableListView;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
public class MainActivity extends Activity
{
    private TextView text  =  null;
    private int[] image  =  { R.drawable.po1,R.drawable.po2,
            R.drawable.po3,R.drawable.po5 };
    private String[] item  =  { "杀生丸","宇智波佐助 ","江户川柯南","樱木花道" };
    private String[][] ability  =  { { "会必杀技","会斗鬼神" },
```

```
            { "会变身术","会分身术","会替身术","会火遁风仙火之术" },{ "会侦探","会玩球
术" },{ "隐藏技术高" } };
    private ExpandableListView explandListView;
    /** 第 1 次调用 activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //通过 ID 查找到 main.xml 中的 TextView 控件
        text = (TextView) findViewById(R.id.text);
        //通过 ID 查找到 main.xml 中的 ExpandableListView 控件
      explandListView = (ExpandableListView) findViewById(R.id.expandableListView);
        //设置 ExpandableListView 适配器
        ExpandableListAdapter adapter = new BaseExpandableListAdapter()
        {
            //处理子项目的单击事件
            @Override
            public boolean isChildSelectable(int groupPosition,int childPosition)
            {
                String str = item[groupPosition]
                        + ability[groupPosition][childPosition];
                updateText(str);
                return true;
            }
            @Override
            public boolean hasStableIds()
            {
                return true;
            }
            //返回父项目的视图控件
            @Override
            public View getGroupView(int groupPosition,boolean isExpanded,
                    View convertView,ViewGroup parent)
            {
                //新建一个线性布局
                LinearLayout ll = new LinearLayout(MainActivity.this);
                //设置布局样式为 Horizontal
                ll.setOrientation(0);
                //设置布局左边距为 50 像素
                ll.setPadding(50,0,0,0);
                //新建一个 ImageView 对象
                ImageView imageView = new ImageView(MainActivity.this);
                //设置 ImageView 要显示的对象 ID
                imageView.setImageResource(image[groupPosition]);
                //将 ImageView 加到线性布局中
                ll.addView(imageView);
                //使用自定义文本框
                TextView textView = getTextView();
                //设置文本框里显示内容
                textView.setText(getGroup(groupPosition).toString());
                //将 TextView 加到线性布局中
                ll.addView(textView);
                return ll;
```

```
        }
        //返回父控件的 ID
        @Override
        public long getGroupId(int groupPosition)
        {
            return groupPosition;
        }
        //返回父控件的总数
        @Override
        public int getGroupCount()
        {
            return ability.length;
        }
        //取得父控件对象
        @Override
        public Object getGroup(int groupPosition)
        {
            return item[groupPosition];
        }
        //取得子控件的数量
        @Override
        public int getChildrenCount(int groupPosition)
        {
            return ability[groupPosition].length;
        }
        //取得子控件的视图
        @Override
        public View getChildView(int groupPosition, int childPosition, boolean isLastChild, View
convertView, ViewGroup parent)
        {
            //使用自定义 TextView 控件
            TextView textView = getTextView();
            //设置自定义 TextView 控件的内容
            textView.setText(getChild(groupPosition, childPosition)
                    .toString());
            return textView;
        }
        //取得子控件的 ID
        @Override
        public long getChildId(int groupPosition, int childPosition)
        {
            return childPosition;
        }
        //取得子控件的对象
        @Override
        public Object getChild(int groupPosition, int childPosition)
        {
            return ability[groupPosition][childPosition];
        }
      //自定义文本框
        public TextView getTextView()
        {
            AbsListView.LayoutParams lp = new AbsListView.LayoutParams(
                    ViewGroup.LayoutParams.FILL_PARENT, 64);
```

```
                TextView textView = new TextView(MainActivity.this);
                textView.setLayoutParams(lp);
                textView.setPadding(20,0,0,0);
                //设置 TextView 控件为向左,水平居中对齐
                textView.setGravity(Gravity.CENTER_VERTICAL | Gravity.LEFT);
                return textView;
            }
        };
        explandListView.setAdapter(adapter);
    }
  private void updateText(String string)
    {
        //将文本信息设置给 TextView 控件显示出来
        text.setText(string);
    }
}
```

运行程序,默认效果如图 5-5(a)所示,当单击相关图片时,即显示相关的信息,如图 5-5(b)所示。



(a) 默认界面



(b) 浏览图像及相关信息

图 5-5　ExpandableListView 用法

5.4　Android 图像切换器

图像切换器使用 ImageSwitcher 表示,用于实现类似于 Windows 操作系统下"Windows 照片查看器"中的上一张、下一张切换图片的功能。在使用 ImageSwitcher 时,必须实现 ViewSwitcher.ViewFactory 接口,并通过 makeView()方法来创建用于显示图片的 ImageView。makeView()方法将返回一个显示图片的 ImageView。在使用图像切换器时,还有一个方法非常重要,那就是 setImageResource()方法,该方法用于指定要在 ImageSwitcher 中显示的图片资源。

下面通过一个实例来说明图像切换器的具体用法。

【例 5-5】 利用 ImageSwitcher 控件实现图像浏览器。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用程序，命名为 ImageSwitcher。

(2) 打开 res\layout 目录下的 main. xml 布局文件，在文件中声明一个 FrameLayout 布局、一个 RelativeLayout 布局、一个 LinearLayout 布局及一个 ImageSwitcher 控件。代码为：

```
<?xml version = "1.0" encoding = "UTF - 8"?>
< FrameLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = " # aabbcc">
    < ImageSwitcher
        android:id = "@ + id/imageSwitcher1"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"/>
    < RelativeLayout
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:orientation = "vertical" >
        < LinearLayout
            android:id = "@ + id/viewGroup"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:layout_alignParentBottom = "true"
            android:layout_marginBottom = "30dp"
            android:gravity = "center_horizontal"
            android:orientation = "horizontal" >
        </LinearLayout >
    </RelativeLayout >
</FrameLayout >
```

(3) 打开 src\fs. imageswitcher 包下的 MainActivity. java 文件，在文件中实现图像的浏览。代码为：

```
package fs.imageswitcher;
import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.view.ViewGroup;
import android.view.animation.AnimationUtils;
import android.widget.ImageSwitcher;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.RelativeLayout.LayoutParams;
import android.widget.Toast;
import android.widget.ViewSwitcher.ViewFactory;
public class MainActivity extends Activity implements ViewFactory,OnTouchListener{
    /**
     * ImagaSwitcher 的引用
```

```
     */
    private ImageSwitcher mImageSwitcher;
    /**
     * 图片 id 数组
     */
    private int[] imgIds;
    /**
     * 当前选中的图片 id 序号
     */
    private int currentPosition;
    /**
     * 按下点的 X 坐标
     */
    private float downX;
    /**
     * 装载点点的容器
     */
    private LinearLayout linearLayout;
    /**
     * 点点数组
     */
    private ImageView[] tips;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        imgIds = new int[]{R.drawable.a01,R.drawable.a02,R.drawable.a03,R.drawable.a04,
                R.drawable.a05};
        //实例化 ImageSwitcher
        mImageSwitcher  = (ImageSwitcher) findViewById(R.id.imageSwitcher1);
        //设置 Factory
        mImageSwitcher.setFactory(this);
        //设置 OnTouchListener,通过 Touch 事件来切换图片
        mImageSwitcher.setOnTouchListener(this);
        linearLayout = (LinearLayout) findViewById(R.id.viewGroup);
        tips = new ImageView[imgIds.length];
        for(int i = 0; i < imgIds.length; i++){
            ImageView mImageView = new ImageView(this);
            tips[i] = mImageView;
            LinearLayout.LayoutParams layoutParams = new LinearLayout.LayoutParams ( new
ViewGroup.LayoutParams(LayoutParams.WRAP_CONTENT,
                    LayoutParams.WRAP_CONTENT));
            layoutParams.rightMargin = 3;
            layoutParams.leftMargin = 3;
        }
        //从上一个界面 GridView 传过来
        currentPosition = getIntent().getIntExtra("position",0);
        mImageSwitcher.setImageResource(imgIds[currentPosition]);
    }
    @Override
    public View makeView() {
        final ImageView i = new ImageView(this);
        i.setBackgroundColor(0xff000000);
        i.setScaleType(ImageView.ScaleType.CENTER_CROP);
```

```
            i.setLayoutParams(new ImageSwitcher.LayoutParams(LayoutParams.FILL_PARENT,LayoutParams.
FILL_PARENT));
            return i ;
        }
        @Override
        public boolean onTouch(View v,MotionEvent event) {
            switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:{
                //手指按下的 X 坐标
                downX = event.getX();
                break;
            }
            case MotionEvent.ACTION_UP:{
                float lastX = event.getX();
                //抬起的时候的 X 坐标大于按下的时候就显示上一张图片
                if(lastX > downX){
                    if(currentPosition > 0){
                    //设置动画
mImageSwitcher.setInAnimation(AnimationUtils.loadAnimation(getApplication(),R.anim.left_
in));
mImageSwitcher.setOutAnimation(AnimationUtils.loadAnimation(getApplication(),R.anim.right_
out));
                    currentPosition --;
    mImageSwitcher.setImageResource(imgIds[currentPosition % imgIds.length]);
                    }else{
                        Toast.makeText(getApplication(),"已经是第 1 张",Toast.LENGTH_SHORT).show();
                    }
                }
                if(lastX < downX){
                    if(currentPosition < imgIds.length - 1){
mImageSwitcher.setInAnimation(AnimationUtils.loadAnimation(getApplication(),R.anim.right_in));
mImageSwitcher.setOutAnimation(AnimationUtils.loadAnimation(getApplication(),R.anim.left_
out));
                    currentPosition ++;
                        mImageSwitcher.setImageResource(imgIds[currentPosition]);
                    }else{
                        Toast.makeText(getApplication(),"到了最后 1 张",Toast.LENGTH_SHORT).show();
                    }
                }
                }
                break;
            }
            return true;
        }
}
```

(4) 在 res 文件夹下创建一个 anim 文件夹，在该文件夹中新建几个文件，用于实现动画的切换，分别命名为 left_in.xml、left_out.xml、right_in.xml 及 right_out.xml，代码分别为：

left_in.xml 文件代码为：

```
<?xml version = "1.0" encoding = "UTF - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android">
    < translate
        android:fromXDelta = " - 100 % p"
```

```
        android:toXDelta = "0"
        android:duration = "500"/>
</set>
```

right_in. xml 文件代码为：

```
<?xml version = "1.0" encoding = "UTF - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android">
    < translate
        android:fromXDelta = "100 % p"
        android:toXDelta = "0"
        android:duration = "500"/>
</set>
```

right_out. xml 文件代码为：

```
<?xml version = "1.0" encoding = "UTF - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android">
    < translate
        android:fromXDelta = "0"
        android:toXDelta = "100 % p"
        android:duration = "500"/>
</set>
```

right_out. xml 文件代码为：

```
<?xml version = "1.0" encoding = "UTF - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android">
    < translate
        android:fromXDelta = "0"
        android:toXDelta = "100 % p"
        android:duration = "500"/>
</set>
```

运行程序，默认界面如图 5-6(a)所示，当拖动鼠标时，即实现图像的切换，切换到最后一张图像效果如图 5-6(b)所示。




(a) 默认图像



(b) 最后1张图像

图 5-6　图像的切换

5.5　Android 画廊视图

画廊视图使用 Gallery 表示，能够按水平方向显示内容，并且可用手指直接拖动图片移动，一般用来浏览图片，被选中的选项位于中间，并且可以响应事件显示信息。在使用画廊视图时，首先需要在屏幕上添加 Gallery 组件，通常使用＜Gallery＞标记在 XML 布局文件中添加。在 XML 布局文件中添加画廊视图的基本格式为：

```
<Gallery
    android:id = "@ + id/gallery1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_below = "@ + id/textView1"
    android:layout_centerHorizontal = "true"
    android:layout_marginTop = "109dp" />
```

Gallery 组件支持的 XML 属性如表 5-4 所示。

表 5-4　Gallery 常用的 XML 属性及描述

XML 属性	方　　法	描　　述
android:animationDuration	setAnimationDuration(int)	设置列表项切换时的动画持续时间
android:gravity	setGravity(int)	设置对齐方式
android:spacing	setSpacing(int)	设置 Gallery 内列表项之间的间距
android:unselectedAlpah	setUnselectedAlpha(float)	设置没有选中的列表项的透明度

Gallery 本身的用法非常简单——基本上与 Spinner 的用法形似，只要为它提供一个内容 Adapter 即可，该 Adapter 的 getView 方法所返回的 View 将作为 Gallery 列表的列表项；如果程序需要监控到 Gallery 选择项的改变，可以通过为 Gallery 添加 onItemSelectedListener 监听器即可实现。

下面通过一个具体实例来演示 Gallery 的用法。

【例 5-6】 应用画廊视图和图像切换器实现幻灯片式图片浏览器。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Gallery_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明一个 ImageSwitcher 控件及一个 Gallery 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:gravity = "center_horizontal"
    android:id = "@ + id/llayout"
    android:background = "#aabbcc">
    <ImageSwitcher
        android:id = "@ + id/imageSwitcher1"
        android:layout_weight = "2"
```

```
        android:paddingTop = "10dp"
        android:paddingBottom = "5dp"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content" >
    </ImageSwitcher>
    <Gallery
        android:id = "@ + id/gallery1"
        android:spacing = "5dp"
        android:layout_weight = "1"
        android:unselectedAlpha = "0.6"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content" />
</LinearLayout>
```

(3) 打开 src/fs.gallery_test 包下的 MainActivity.java 文件，在文件中实现幻灯片式图像浏览器。代码为：

```
package fs.gallery_test;
import android.app.Activity;
import android.content.res.TypedArray;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewGroup.LayoutParams;
import android.view.animation.AnimationUtils;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageSwitcher;
import android.widget.ImageView;
import android.widget.ViewSwitcher.ViewFactory;
public class MainActivity extends Activity {
    private int[] imageId = new int[] { R.drawable.a01,R.drawable.a02,
R.drawable.a03,R.drawable.a04,R.drawable.a05,};          //定义并初始化保存图片 id 的数组
    private ImageSwitcher imageSwitcher;                 //声明一个图像切换器对象
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Gallery gallery = (Gallery) findViewById(R.id.gallery1);//获取 Gallery 组件
        //获取图像切换器
        imageSwitcher = (ImageSwitcher) findViewById(R.id.imageSwitcher1);
                //设置动画效果
                imageSwitcher.setInAnimation(AnimationUtils.loadAnimation(this,
                        android.R.anim.fade_in));               //设置淡入动画
                imageSwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,
                        android.R.anim.fade_out));              //设置淡出动画
                imageSwitcher.setFactory(new ViewFactory() {
                    @Override
                    public View makeView() {
```

```
//实例化一个 ImageView 类的对象
        ImageView imageView = new ImageView(MainActivity.this);
//设置保持纵横比居中缩放图像
imageView.setScaleType(ImageView.ScaleType.FIT_CENTER);
imageView.setLayoutParams(new ImageSwitcher.LayoutParams(
        LayoutParams.WRAP_CONTENT,LayoutParams.WRAP_CONTENT));
                return imageView;                    //返回 imageView 对象
                }
            });
        //使用 BaseAdapter 指定要显示的内容
        BaseAdapter adapter = new BaseAdapter() {
            @Override
            public View getView(int position,View convertView,ViewGroup parent) {
                ImageView imageview;                 //声明 ImageView 的对象
                if (convertView == null) {
    imageview = new ImageView(MainActivity.this);    //实例化 ImageView 的对象
        imageview.setScaleType(ImageView.ScaleType.FIT_XY);  //设置缩放方式
                    imageview.setPadding(5,0,5,0);   //设置 ImageView 的内边距
                } else {
                    imageview = (ImageView) convertView;
                }
imageview.setImageResource(imageId[position]);    //为 ImageView 设置要显示的图片
                return imageview;                    //返回 ImageView
            }
            //功能：获得当前选项的 ID
            @Override
            public long getItemId(int position) {
                return position;
            }
            //功能：获得当前选项
            @Override
            public Object getItem(int position) {
                return position;
            }
            //获得数量
            @Override
            public int getCount() {
                return imageId.length;
            }
        };
        gallery.setAdapter(adapter);                 //将适配器与 Gallery 关联
        gallery.setSelection(imageId.length / 2);    //让中间的图片选中
        gallery.setOnItemSelectedListener(new OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent,View view,
                    int position,long id) {
            imageSwitcher.setImageResource(imageId[position]);  //显示选中的图片
            }
            @Override
            public void onNothingSelected(AdapterView<?> arg0) {
```

```
            }
        });
    }
}
```

运行程序,效果如图 5-7 所示。



图 5-7　图像的幻灯片式切换

5.6　Android 网页浏览视图

在 Android 手机中内置了一款高性能 WebKit 内核浏览器,WebView 组件就是由 Webkit 封装而来的,可以用它来显示一个 Web 页面。

WebView 是一个浏览器控件,通过这个控件可以直接访问网页,或者把输入的 HTML 字符串显示出来,功能比较强大,有以下几个优点:

- 功能强大,支持 CSS、JavaScript 等 HTML 语言,这样页面就能更漂亮。
- 能够对浏览器控件进行非常详细的设置,例如字体大小、背景色、滚动条样式等。
- 能够捕捉到所有浏览器操作,例如单击 URL、打开或关闭 URL。
- 能够很好地融入布局。
- 甚至 WebView 还能和 JS 进行交互。

WebView 使用了 WebKit 渲染引擎加载显示网页,实现 WebView 有以下两种不同的方法:

第 1 种方法的步骤:

(1) 要在 Activity 中实例化 WebView 组件:WebView webView = new WebView(this);。

(2) 调用 WebView 的 loadUrl()方法,设置 WebView 要显示的网页:

- 互联网用:webView. loadUrl("http://www. google. com");。
- 本地文件用:webView. loadUrl("file:///android_asset/XX. html");本地文件存放在 assets 文件中。

(3) 调用 Activity 的 setContentView()方法来显示网页视图。

(4) 用 WebView 单击链接看了很多页面以后为了让 WebView 支持回退功能,需要覆盖 Activity 类的 onKeyDown()方法,如果不做任何处理,单击系统回退键,整个浏览器会调用 finish()而结束自身,而不是回退到上一页面。

(5) 需要在 AndroidManifest. xml 文件中添加权限,否则会出现 Web page not available 错误。

```
<uses-permission android:name="android.permission.INTERNET" />
```

下面通过一个实例来演示 WebView 控件第一种方法的用法。

【例 5-7】 使用 WebView 控件的第一种方法实现一个浏览器。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 WebView_test1。

(2) 打开 src\fs. webview_test1 包下的 MainActivity. java 文件。代码为:

```
package fs.webview_test1;
import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.webkit.WebView;
public class MainActivity extends Activity {
    private WebView webview;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //实例化 WebView 对象
        webview = new WebView(this);
        //设置 WebView 属性,能够执行 JavaScript 脚本
        webview.getSettings().setJavaScriptEnabled(true);
        //加载需要显示的网页
        webview.loadUrl("http://www.hao123.cn/");
        //设置 Web 视图
        setContentView(webview);
    }
    @Override
    //设置回退
    //覆盖 Activity 类的 onKeyDown(int keyCoder,KeyEvent event)方法
    public boolean onKeyDown(int keyCode,KeyEvent event) {
        if ((keyCode == KeyEvent.KEYCODE_BACK) && webview.canGoBack()) {
            webview.goBack();                    //goBack()表示返回 WebView 的上一页面
            return true;
        }
        return false;
    }
}
```

(3) 打开 AndroidManifest.xml 文件,在文件中设置权限。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.webview_test1"
    android:versionCode = "1"
    android:versionName = "1.0" >
    < uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
    < application
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        < activity
            android:name = "fs.webview_test1.MainActivity"
            android:label = "@string/app_name" >
            < intent - filter >
                < action android:name = "android.intent.action.MAIN" />
                < category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter >
        </activity >
    </application >
    <! -- 权限设置   -- >
< uses - permission android:name = "android.permission.INTERNET"/>
</manifest >
```

运行程序,效果如图 5-8 所示。



图 5-8　网页浏览视图 1

第 2 种方法的步骤：

(1) 在布局文件中声明 WebView。

(2) 在 Activity 中实例化 WebView。

(3) 调用 WebView 的 loadUrl()方法，设置 WebView 要显示的网页。

(4) 为了让 WebView 能够响应超链接功能，调用 setWebViewClient()方法，设置 WebView 视图。

(5) 用 WebView 单击链接看了很多页面以后为了让 WebView 支持回退功能，需要覆盖 Activity 类的 onKeyDown()方法，如果不做任何处理，单击系统回退键，整个浏览器会调用 finish()而结束自身，而不是回退到上一页面。

(6) 需要在 AndroidManifest.xml 文件中添加权限，否则出现 Web page not available 错误。

```
<uses-permission android:name="android.permission.INTERNET"/>
```

下面通过一个实例来演示 WebView 控件第 2 种方法的用法。

【例 5-8】 使用 WebView 控件的第 1 种方法实现一个浏览器。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 WebView_test2。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中声明一个 WebView 控件。代码为：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#aabbcc">
    <WebView
        android:id="@+id/webview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"/>
</LinearLayout>
```

(3) 打开 src\fs.webview_test2 包下的 MainActivity.java 文件。代码为：

```
package fs.webview_test2;
import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.webkit.WebView;
import android.webkit.WebViewClient;
public class MainActivity extends Activity {
    private WebView webview;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        webview = (WebView) findViewById(R.id.webview);
        //设置 WebView 属性,能够执行 JavaScript 脚本
        webview.getSettings().setJavaScriptEnabled(true);
```

```
        //加载需要显示的网页
        webview.loadUrl("http://www.hao123.cn/");
        //设置 Web 视图
        webview.setWebViewClient(new HelloWebViewClient ());
    }
    @Override
    //设置回退
    //覆盖 Activity 类的 onKeyDown(int keyCoder,KeyEvent event)方法
    public boolean onKeyDown(int keyCode,KeyEvent event) {
        if ((keyCode == KeyEvent.KEYCODE_BACK) && webview.canGoBack()) {
            webview.goBack();                    //goBack()表示返回 WebView 的上一页面
            return true;
        }
        return false;
    }
    //Web 视图
    private class HelloWebViewClient extends WebViewClient {
        @Override
        public boolean shouldOverrideUrlLoading(WebView view,String url) {
            view.loadUrl(url);
            return true;
        }
    }
}
```

(4) 打开 AndroidManifest.xml 文件,在文件中设置权限。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.webview_test2"
    android:versionCode = "1"
    android:versionName = "1.0" >
    < uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
    < application
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        < activity
            android:name = "fs.webview_test2.MainActivity"
            android:label = "@string/app_name" >
            < intent - filter >
                < action android:name = "android.intent.action.MAIN" />
                < category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter >
        </activity >
    </application >
    <! -- 设置权限 -- >
< uses - permission android:name = "android.permission.INTERNET"/>
</manifest >
```

运行程序,效果如图 5-9 所示。



图 5-9　网页浏览视图 2

5.7　Android 多页视图

ViewPager 用于实现多面页的切换效果，该类存在于 Google 的兼容包里面，所以在引用时记住在 BuilldPath 中加入“android-support-v4.jar”。

ViewPager 非常适合用于实现多页面的滑动切换效果，相同的多页面切换可以是 TabHost，但是 Tabhost 标题栏需要重写，稍微麻烦一点。所以采用 ViewPager 来实现滑动切换，网上很多都是使用 ViewPager 来加载 View 做一些静态的页面展示，像导航、图片展示、使用教程等。

下面通过一个实例来演示 ViewPager 控件的用法。

【例 5-9】 利用 ViewPage 控件浏览图片。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 ViewPage_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中定义一个 ViewPager 控件及一个 PagerTitleStrip 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = " # aabbcc">
< android.support.v4.view.ViewPager
        android:id = "@ + id/viewpager"
```

```
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_gravity = "center" >
<android.support.v4.view.PagerTitleStrip
        android:id = "@ + id/pagertitle"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_gravity = "top" />
    </android.support.v4.view.ViewPager>
</LinearLayout>
```

(3) 打开 src\fs.lviewpage_test 包下的 MainActivity.java 文件，用于实现图像的翻页浏览功能。代码为：

```
package fs.viewpage_test;
import java.util.ArrayList;
import android.os.Bundle;
import android.app.Activity;
import android.graphics.drawable.Drawable;
import android.support.v4.view.PagerAdapter;
import android.support.v4.view.PagerTitleStrip;
import android.support.v4.view.ViewPager;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.View;
import android.widget.ImageView;
import android.widget.LinearLayout;
public class MainActivity extends Activity {
    /** 第 1 次调用 activity 活动 */
    private ViewPager mViewPager;
    private PagerTitleStrip mPagerTitleStrip;
    private int[] pics = { R.drawable.g1,R.drawable.g2,R.drawable.g4 };
    final ArrayList<View> views = new ArrayList<View>();
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mViewPager = (ViewPager) findViewById(R.id.viewpager);
        mPagerTitleStrip = (PagerTitleStrip) findViewById(R.id.pagertitle);
        LinearLayout.LayoutParams mParams = new LinearLayout.LayoutParams(
                LinearLayout.LayoutParams.WRAP_CONTENT,
                LinearLayout.LayoutParams.WRAP_CONTENT);
        //将要分页显示的 View 装入数组中
        for (int i = 0; i < pics.length; i++) {
            ImageView iv = new ImageView(this);
            iv.setLayoutParams(mParams);
            iv.setImageResource(pics[i]);
            views.add(iv);
        }
        //每个页面的 Title 数据
        final ArrayList<String> titles = new ArrayList<String>();
        titles.add("tab1");
        titles.add("tab2");
        titles.add("tab3");
```

```
            //填充 ViewPager 的数据适配器
            PagerAdapter mPagerAdapter = new PagerAdapter() {
                @Override
                public boolean isViewFromObject(View arg0,Object arg1) {
                    return arg0 == arg1;
                }
                @Override
                public int getCount() {
                    return views.size();
                }
                @Override
                public void destroyItem(View container,int position,Object object) {
                    ((ViewPager) container).removeView(views.get(position));
                }
                @Override
                public CharSequence getPageTitle(int position) {
                    return titles.get(position);
                }
                @Override
                public Object instantiateItem(View container,int position) {
                    ((ViewPager) container).addView(views.get(position));
                    return views.get(position);
                }
            };
            mViewPager.setAdapter(mPagerAdapter);
        }
    @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            getMenuInflater().inflate(R.menu.main,menu);
            return true;
        }
    }
```

运行程序，效果如图 5-10 所示。



图 5-10　图片的翻页浏览

5.8 Android 切换列表

TabActivity(切换列表)继承自 Activity，其内部定义好了 TabHost，可以通过 getTabHost()获取。TabHost 包含了两种子元素：一些可以自由选择的 Tab 和 Tab 对应的内容 TabContentto，在 Layout 的<TabHost>下，它们分别对应 TabWidget 和 FrameLayout。

在 TabActivity 中，只在第 1 次进入时经历了 onCreate()、onStart()、onResume() 3 个阶段，然后在退出该页面时经历了 onPause()、onStop()和 onDestroy()2 个阶段。其他时间无论其中的子 Activity 如何切换，都不会再进入 TabActivity 的生命周期。

而子 Activity，在第 1 次创建的时候，都会经历 onCreate()、onStart()、onResume() 3 个阶段，期间在各子 Activity 中切换，经历了 onPause()和 onResume()两个阶段，然后在主 TabActivity 退出时经历了 onPause()、onStop()和 onDestroy() 3 个阶段。

下面通过一个实例来演示 TabActivity 控件的用法。

【例 5-10】 主要用于实现标签的切换功能。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 TabActivity_test。

(2) 打开 res\layout 目录下的 main. xml 文件，在文件中实现 3 个标签页面。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<FrameLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical" android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
<LinearLayout
    android:id = "@ + id/tabFood"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
    <TextView
        android:id = "@ + id/TextView01"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"/>
</LinearLayout>
<LinearLayout
    android:id = "@ + id/tabCloths"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:background = "#aabbcc">
<TextView
        android:id = "@ + id/TextView02"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"/>
    </LinearLayout>
<LinearLayout
        android:id = "@ + id/tabOutside"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:background = "#aabbcc">
<TextView
        android:id = "@ + id/TextView03"
        android:layout_width = "wrap_content"
```

```
            android:layout_height = "wrap_content"/>
        </LinearLayout>
    </FrameLayout>
```


(3) 打开 src/fs. tabactivity_test 包下的 MainActivity. java 文件，在文件中实现 3 个标签的切换，当运行程序时，弹出一个提示框。代码为：

```
package fs.tabactivity_test;
import android.app.AlertDialog;
import android.app.TabActivity;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.widget.TabHost;
import android.widget.TextView;
import android.widget.TabHost.OnTabChangeListener;
import android.widget.TabHost.TabSpec;
public class MainActivity extends TabActivity implements OnTabChangeListener {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        new AlertDialog.Builder(this)
        .setTitle("冬季小常识")
        .setMessage("如何快乐过冬!")
        .setPositiveButton("确定",null)
        .show();
        TabHost tabHost = this.getTabHost();
        LayoutInflater.from(this).inflate(R.layout.main,
                tabHost.getTabContentView(),true);
        TabSpec tabFood = tabHost.newTabSpec("food").setIndicator("饮食",
                this.getResources().getDrawable(R.drawable.food)).setContent(
                R.id.tabFood);
        tabHost.addTab(tabFood);
        TabSpec tabCloths = tabHost.newTabSpec("cloths").setIndicator("保暖",
                this.getResources().getDrawable(R.drawable.b6)).setContent(
                R.id.tabCloths);
        tabHost.addTab(tabCloths);
        TabSpec tabOutside = tabHost.newTabSpec("outside").setIndicator("出行",
                this.getResources().getDrawable(R.drawable.tu)).setContent(
                R.id.tabOutside);
        tabHost.addTab(tabOutside);
        tabHost.setOnTabChangedListener(this);
        onTabChanged("food");
    }
    public void onTabChanged(String tabId) {
        if(tabId.equals("food"))
        {
            TextView tv = (TextView)findViewById(R.id.TextView01);
            tv.setText(
           "1.怕冷与饮食中缺少无机盐有关,应多摄取含根茎的食物,如山芋、藕、大葱、土豆等。"
+ "\n" +
    "2.冬季保健应适当吃"冷",常饮凉白开水有预防感冒之功效." +
      "平时要多饮水,以维持水代谢平衡,防止皮肤干裂,邪火上侵。" + "\n" +
    "3.香菇味道鲜美且具有防治流感的作用,常吃还能阻止血管硬化。" + "\n" +
      "4.多吃蔬菜、水果,如葡萄、萝卜、梨、柿、莲子、百合、甘蔗、菠萝、香蕉等," +
"以补充体内维生素和矿物质,中和体内多余的酸性代谢物,起到清火解毒润肺之效; " +
"多吃豆类等高蛋白植物性食物,少吃油腻辛辣食物,不宜多吃烧烤。" + "\n" +
```

```
"5.冬天适当吃点凉菜有利于减肥."迫使"身体自身取暖,多消耗一些脂肪。" + "\n" +
                    "6.在冬季不要经常饮用一些过热的饮料。温度过高的饮料可造成广泛的皮
肤粘膜损伤," +
                    "蛋白质在 43 度开始变性,胃肠道粘液在达到 60 度会产生不可逆的降解,所
以不要饮用过热的饮品。" + "\n"
                    );
        }
        if(tabId.equals("cloths"))
        {
            TextView tv2 = (TextView)findViewById(R.id.TextView02);
            tv2.setText(
                    "1.衣服选择保暖、舒适冬; " + "\n" +
                    "2.根据室温控制穿衣; " + "\n" +
                    "3.穿衣忌衣领过高过; " + "\n" +
                    "4.小孩穿衣宜少不宜多"
                    );
        }
        if(tabId.equals("outside"))
        {
            TextView tv3 = (TextView)findViewById(R.id.TextView03);
            tv3.setText(
                    "1. 冬季出游必备物品" + "\n" +
                    "(1) 防寒衣; " + "\n" +
                    "(2) 不怕凉的食物; " + "\n" +
                    "2. 冬季出行,注意保暖" + "\n" +
                    "(1) 应该选择防风性的外套; " + "\n" +
                    "(2) 应该注意饮食的搭配; " + "\n" +
                    "(3) 适量运动; " + "\n"
                    );
        }
    }
}
```

运行程序,默认界面如图 5-11(a)所示,当选择其他标签时,效果如图 5-11(b)所示。





(a) 默认界面



(b) 切换标签界面

图 5-11　切换列表控件使用

5.9 Android 滑动式抽屉

SlidingDrawer 隐藏屏外的内容，并允许用户通过 handle 以显示隐藏的内容。它可以垂直或水平滑动，它由两个 View 组成，其一是可以拖动的 handle，其二是隐藏内容的 View。它里面的控件必须设置布局，在布局文件中必须指定 handle 和 content。

其重要的属性主要有：

- android:allowSingleTap：指示是否可以通过 handle 打开或关闭。
- android:animateOnClick：指示是否当使用者按下手柄打开/关闭时是否该有一个动画。
- android:content：隐藏的内容。
- android:handle：手柄。

其常用的重要方法主要有：

- animateClose()：关闭时实现动画。
- close()：即时关闭。
- getContent()：获取内容。
- isMoving()：指示 SlidingDrawer 是否在移动。
- isOpened()：指示 SlidingDrawer 是否已全部打开。
- lock()：屏蔽触摸事件。
- setOnDrawerCloseListener(SlidingDrawer. OnDrawerCloseListener onDrawerCloseListener)：SlidingDrawer 关闭时调用。
- unlock()：解除屏蔽触摸事件。
- toggle()：切换打开和关闭的抽屉 SlidingDrawer。

下面通过一个实例来演示 SlidingDrawer 控件的使用。

【例 5-11】 下面实现一个滑动式抽屉。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 SlidingDrawer_test。

(2) 打开 res\layout 目录下的 main. xml 文件，在文件中声明一个 SlidingDrawer 控件、一个 TextView 控件及一个 ImageButton 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout
     xmlns:android = "http://schemas.android.com/apk/res/android"
       android:layout_width = "fill_parent"
       android:layout_height = "fill_parent"
       android:orientation = "vertical"
       android:background = "@drawable/kp">
< SlidingDrawer
       android:layout_width = "fill_parent"
       android:layout_height = "fill_parent"
       android:handle = "@ + id/handle"
       android:content = "@ + id/content"
       android:orientation = "vertical"
       android:id = "@ + id/slidingdrawer">
```

```
<ImageButton
        android:id="@id/handle"
        android:layout_width="50dip"
        android:layout_height="44dip"
        android:src="@drawable/up" />
<LinearLayout
        android:id="@id/content"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="#aabbcc">
    <TextView
          android:text="这是一个滑动式抽屉的示例"
          android:id="@+id/tv"
          android:textSize="18px"
          android:textColor="#000000"
          android:gravity="center_vertical|center_horizontal"
          android:layout_width="match_parent"
          android:textStyle="bold"
          android:layout_height="match_parent"/>
  </LinearLayout>
 </SlidingDrawer>
</LinearLayout>
```

(3) 打开 src\fs.slidingdrawer_test 包下的 MainActivity.java 文件,在文件中实现滑动式抽屉。代码为:

```
package fs.slidingdrawer_test;
import android.app.Activity;
import android.os.Bundle;
import android.widget.ImageButton;
import android.widget.SlidingDrawer;
import android.widget.TextView;
public class MainActivity extends Activity {
    private SlidingDrawer mDrawer;
    private ImageButton imbg;
    private Boolean flag=false;
    private TextView tv;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //TODO 自动存根法
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        imbg=(ImageButton)findViewById(R.id.handle);
        mDrawer=(SlidingDrawer)findViewById(R.id.slidingdrawer);
        tv=(TextView)findViewById(R.id.tv);
        mDrawer.setOnDrawerOpenListener(new SlidingDrawer.OnDrawerOpenListener()
        {
            @Override
            public void onDrawerOpened() {
                flag=true;
                imbg.setImageResource(R.drawable.down);
```

```
                }
            });
        mDrawer.setOnDrawerCloseListener(new SlidingDrawer.OnDrawerCloseListener(){
            @Override
            public void onDrawerClosed() {
                flag = false;
                imbg.setImageResource(R.drawable.up);
                }
            });
        mDrawer.setOnDrawerScrollListener(new SlidingDrawer.OnDrawerScrollListener(){
            @Override
            public void onScrollEnded() {
                tv.setText("结束拖动");
                }
            @Override
    public void onScrollStarted() {
                tv.setText("开始拖动");
                }
            });
        }
    }
```

运行程序,默认效果如图 5-12(a)所示,单击界面中的向上图标按钮,则切换到另一个页面,效果如图 5-12(b)所示。




(a) 默认界面



(b) 切换界面

图 5-12　滑动抽屉

5.10　Android 点阵图像

Bitmap 称为点阵图像或绘制图像,是由称作像素(图片元素)的单个点组成的,这些点通过不同的排列和染色以构成图样。Bitmap 是 Android 系统中图像处理最重要的类之一,

用它可以获取图像文件信息，对图像进行剪切、旋转、缩放等操作，并可以将图像保存成特定格式的文件。Bitmap 位于 android. graphics 包中，Bitmap 不提供对外的构造方法，只能通过 BitmapFactory 类进行实例化。利用 BitmapFactory 的 decodeFile 方法可以从特定文件中获取 Bitmap 对象，也可以使用 decodeResource()从特定的图片资源中获取 Bitmap 对象。

下面通过一个实例来演示 Bitmap 控件的用法。

【例 5-12】 从资源文件中创建 Bitmap 对象，并进行操作。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Bitmap_test。

(2) 打开 res\layout 目录下的 main. xml 文件，在文件中声明一个 SeekBar 控件及一个 ImageView 控件。代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc">
    <SeekBar
        android:id = "@ + id/seekBar1"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:layout_alignParentTop = "true"
        android:layout_centerHorizontal = "true"
        android:layout_marginTop = "23dp" />
    <ImageView
        android:id = "@ + id/imageView1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/seekBar"
        android:layout_below = "@ + id/seekBar"
        android:layout_marginLeft = "68dp"
        android:layout_marginTop = "112dp"
        android:src = "@drawable/ic_launcher" />
</RelativeLayout>
```

(3) 打开 src\fs. bitmap_test 包下的 MainActivity. java 文件，在文件中实现当拖动滑动条时，实现图像进行旋转。代码为：

```
package fs.bitmap_test;
import android.os.Bundle;
import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Matrix;
import android.view.Menu;
import android.widget.ImageView;
```

```
import android.widget.SeekBar;
public class MainActivity extends Activity {
    ImageView myImageView;
    Bitmap myBmp,newBmp;
    int bmpWidth,bmpHeight;
    SeekBar seekbarRotate;
    float rotAnagle;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        myImageView = (ImageView)findViewById(R.id.imageView1);
        //由 Resource 载入图像
        myBmp = BitmapFactory.decodeResource(getResources(),R.drawable.ic_launcher);
        bmpWidth = myBmp.getWidth();
        bmpHeight = myBmp.getHeight();
        //实例化 maxtrix
        Matrix matrix = new Matrix();
        //设定 Matrix 属性 x、y 缩放比例为 1.5
        matrix.postScale(1.5F,1.5F);
        //顺时针旋转 45°
        matrix.postRotate(45.0F);
        newBmp = Bitmap.createBitmap(myBmp,0,0,bmpWidth,bmpHeight,matrix,true);
        seekbarRotate = (SeekBar)findViewById(R.id.seekBar1);
        seekbarRotate.setOnSeekBarChangeListener(onRotate);
    }
    private SeekBar.OnSeekBarChangeListener onRotate = new SeekBar.OnSeekBarChangeListener()
{
        @Override
        public void onStopTrackingTouch(SeekBar seekBar1) {
            //TODO 自动生成的方法存根
        }
        @Override
        public void onStartTrackingTouch(SeekBar seekBar1) {
            //TODO 自动生成的方法存根
        }
        @Override
    public void onProgressChanged(SeekBar seekBar1,int progress,boolean fromUser) {
            //TODO 自动生成的方法存根
            Matrix m = new Matrix();
            m.postRotate((float)progress * 3.6F);
            newBmp = Bitmap.createBitmap(myBmp,0,0,bmpWidth,bmpHeight,m,true);
            myImageView.setImageBitmap(newBmp);
        }
    };
}
```

运行程序,默认效果如图 5-13(a)所示,当拖动滑动条时,效果如图 5-13(b)所示。



(a) 默认界面



(b) 图像旋转

图 5-13　改变点阵图像

5.11　Android 视图综合实例

在前面章节中已经介绍了实现 Android 视图的几种控件,下面通过一个综合实例来演示 Android 视图的用法。

【例 5-13】 在 Android 中实现仿手机 QQ 登录状态的显示功能。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 QQ_test。

(2) 打开 res\layout 目录下的 main.xml 文件,在文件中添加一个 TableLayout 布局,并在该布局管理器中添加 3 个 TableRow 表格行,接下来在每个表格行中添加用户登录界面相关的控件,最后设置表格的第 1 列和第 4 列允许被拉伸。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< TableLayout android:id = "@ + id/tableLayout1"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:gravity = "center_vertical"
    android:stretchColumns = "0,3"
    android:background = "#aabbcc">
    <! -- 第 1 行 -->
    < TableRow android:id = "@ + id/tableRow1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content">
        < TextView
            android:id = "@ + id/textView1"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "用户名: "
            android:textSize = "24px" />
        < EditText
```

```
                android:id = "@ + id/user"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:ems = "10"
                android:minWidth = "200px"
                android:textSize = "24px" >
                <requestFocus />
            </EditText>
            <TextView />
        </TableRow>
        <!-- 第 2 行 -->
        <TableRow android:id = "@ + id/tableRow2"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content">
            <TextView android:text = "密    码: "
                android:id = "@ + id/textView2"
                android:textSize = "24px"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"/>
            <EditText android:layout_height = "wrap_content"
                android:layout_width = "wrap_content"
                android:textSize = "50px"
                android:id = "@ + id/pwd"
                android:inputType = "textPassword"/>
            <TextView />
        </TableRow>
        <!-- 第 3 行 -->
        <TableRow android:id = "@ + id/tableRow3"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content">
            <Button
                android:text = "登录"
                android:id = "@ + id/button1"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"/>
            <Button
                android:text = "退出"
                android:id = "@ + id/button2"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"/>
            <TextView />
        </TableRow>
    </TableLayout>
```

(3) 在 res\layout 目录下中创建一个 items.xml 文件,在文件中声明一个 LinearLayout 布局,在布局文件中定义一个 ImageView 控件及一个 TextView 控件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout
xmlns:android = "http://schemas.android.com/apk/res/android"
android:orientation = "horizontal"
android:layout_width = "match_parent"
android:layout_height = "match_parent">
<ImageView
```

```
        android:id = "@ + id/image"
        android:paddingLeft = "10px"
        android:paddingTop = "20px"
        android:paddingBottom = "20px"
        android:adjustViewBounds = "true"
        android:maxWidth = "72px"
        android:maxHeight = "72px"
        android:layout_height = "wrap_content"
        android:layout_width = "wrap_content"/>
    < TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:padding = "10px"
        android:layout_gravity = "center"
        android:id = "@ + id/title"/>
    </LinearLayout >
```

(4) 打开 src\fs. qq_test 包下的 MainActivity. java 文件,在文件中实现 QQ 登录、显示登录状态及改变登录状态。代码为:

```
package fs.qq_test;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.AlertDialog.Builder;
import android.app.Notification;
import android.app.NotificationManager;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.SimpleAdapter;
import android.widget.TableRow;
public class MainActivity extends Activity {
    final int NOTIFYID_1 = 123;                                    //第 1 个通知的 ID
    private String user = "匿名";                                  //用户名
    private NotificationManager notificationManager;               //定义通知管理器对象
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //获取通知管理器,用于发送通知
        notificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
        Button button1 = (Button) findViewById(R.id.button1);    //获取"登录"按钮
        //为"登录"按钮添加单击事件监听器
        button1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                EditText etUser = (EditText)findViewById(R.id.user);
```

```
                if(!"".equals(etUser.getText())){
                    user = etUser.getText().toString();
                }
                sendNotification();                                    //发送通知
            }
        });
        Button button2 = (Button) findViewById(R.id.button2);   //获取"退出"按钮
        //为"退出"按钮添加单击事件监听器
        button2.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                notificationManager.cancel(NOTIFYID_1);         //清除通知
//在布局中的第 1 行显示
                ((TableRow)findViewById(R.id.tableRow1)).setVisibility(View.VISIBLE);
//在布局中的第 2 行显示
                ((TableRow)findViewById(R.id.tableRow2)).setVisibility(View.VISIBLE);
//改变"更改登录状态"按钮上显示的文字
          ((Button)findViewById(R.id.button1)).setText("登录");
        }
        });
    }
    //发送通知
    private void sendNotification() {
        Builder builder = new AlertDialog.Builder(MainActivity.this);
        builder.setIcon(R.drawable.c8);                          //设置对话框的图标
        builder.setTitle("登录状态：");                           //设置对话框的标题
        final int[] imageId = new int[] { R.drawable.img1,R.drawable.img2,
            R.drawable.img3,R.drawable.img4 };            //定义并初始化保存图片 id 的数组
        //定义并初始化保存列表项文字的数组
        final String[] title = new String[] { "在线","隐身","忙碌中","离线" };
    //创建一个 list 集合
        List<Map<String,Object>> listItems = new ArrayList<Map<String,Object>>();
                    //通过 for 循环将图片 id 和列表项文字放到 Map 中，并添加到 list 集合中
        for (int i = 0; i < imageId.length; i++) {
Map<String,Object> map = new HashMap<String,Object>();//实例化 map 对象
            map.put("image",imageId[i]);
            map.put("title",title[i]);
            listItems.add(map);                           //将 map 对象添加到 List 集合中
        }
        final SimpleAdapter adapter = new SimpleAdapter(MainActivity.this,
                listItems,R.layout.items,new String[] { "title","image" },
                new int[] { R.id.title,R.id.image });     //创建 SimpleAdapter
        builder.setAdapter(adapter,new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog,int which) {
                Notification notify = new Notification(); //创建一个 Notification 对象
                notify.icon = imageId[which];
                notify.tickerText = title[which];
                notify.when = System.currentTimeMillis(); //设置发送时间
                notify.defaults = Notification.DEFAULT_SOUND;  //设置默认声音
                notify.setLatestEventInfo(MainActivity.this,user,
                        title[which],null);                //设置事件信息
    notificationManager.notify(NOTIFYID_1,notify);        //通知管理器发送通知
//让布局中的第 1 行不显示
```

```
((TableRow)findViewById(R.id.tableRow1)).setVisibility(View.INVISIBLE);
//让布局中的第 2 行不显示
((TableRow)findViewById(R.id.tableRow2)).setVisibility(View.INVISIBLE);
//改变"登录"按钮上显示的文字
                ((Button)findViewById(R.id.button1)).setText("更改状态");
    }
        });
        builder.create().show();                        //创建对话框并显示
    }
}
```

运行程序,将显示一个用户登录界面,如图 5-14(a)所示,输入用户名(bellflower)和密码(111)后,单击"登录"按钮将弹出如图 5-14(b)所示的登录状态列表框,选择对应的登录状态,即在状态栏上显示代表登录状态的图标,单击界面中的"更改状态"按钮,可改变登录状态,单击"退出"按钮,即可删除该通知。



(a) 默认登录界面



(b) 登录状态对话框



(c) 通知显示状态



(d) 更改状态

图 5-14　QQ 登录界面

第6章　Android 动画

在 Android 中除了前面介绍的控件外，还可以进行动画播放。本章将要介绍在 Android 平台下动画播放的技术，在进行用户界面的开发时，除了为控件设置合理的布局和外观外，让控件播放动画或许更能提高用户的体验效果。在 Android 平台下通过简单的开发即可让 View 对象播放指定的动画。

Animations 从总体上可以分为两大类：

- Tweened Animations（补间动画）：类 Animations 提供了旋转、移动、伸展和淡出等效果。
- Frame-by-frame Animations（帧动画）：类 Animations 可以创建一个 Drawable 序列，这些 Drawable 可以按照指定的时间间歇一个一个的显示。

6.1　Android 帧动画

帧(Frame Animation)动画是比较传统的动画方式，帧动画将一系列的图片文件像放电影般依次进行播放，帧动画主要用到的类是 AnimationDrawable，每个帧动画都是一个 AnimationDrawable 对象。

定义帧动画可以在代码中直接进行，也可以通过 XML 文件定义，定义帧动画的 XML 文件将存放在项目的 res\anim 目录下。XML 文件中指定的图片帧出现的顺序及每个帧的持续时间。在帧动画的 XML 文件中主要用到的标记及其属性值如表 6-1 所示。

表 6-1　帧动画中标记及其属性说明

标记名称	属性值	说明
<animation-list>	android:oneshot：如果设置为 true，则该动画只播放一次，然后停止在最后一帧	Frame Animation 的根标记，包含若干<item>标记
<item>	android:drawable：图片帧的引用； android:duration：图片帧的停留时间； android:visible：图片帧是否可见	每个<item>标记定义一个图片帧，其中包含图片资源的引用等属性

定义逐帧动画的 XML 语法格式为：

```
[html] view plaincopyprint?<?xml version = "1.0" encoding = "utf - 8"?>
<animation - list xmlns:android = "http://schemas.android.com/apk/res/android"
    android:oneshot = ["true" | "false"] >
    < item
```

```
        android:drawable = "@[package:]drawable/drawable_resource_name"
        android:duration = "integer" />
</animation - list >
```

下面通过一个实例来演示帧动画的使用效果。

【例 6-1】 在 Android 中实现一个帧动画的播放。其具体实现步骤为：

（1）在 Eclipse 中创建一个 Android 中应用项目，命名为 FrameAnimation_test。

（2）在 res 下创建一个 anim 文件，在文件中新建一个 frame.xml 文件，用于存放帧动画。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< animation - list
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:oneshot = "false">
    < item
        android:drawable = "@drawable/ab"
        android:duration = "100"/>
      < item
          android:drawable = "@drawable/ac"
          android:duration = "100"/>
       < item
           android:drawable = "@drawable/ad"
           android:duration = "100"/>
         < item
             android:drawable = "@drawable/ae"
             android:duration = "100"/>
          < item
              android:drawable = "@drawable/ae"
              android:duration = "100"/>
            < item
                android:drawable = "@drawable/af"
                android:duration = "100"/>
             < item
                 android:drawable = "@drawable/ag"
                 android:duration = "100"/>
</animation - list >
```

（3）打开 res\layout 目录下的 main.xml 文件，在文件中声明两个 Button 控件、两个 RadioButton、一个 RadioGroup 控件、一个 TextView 控件、一个 SeekBar 控件及一个 ImageView 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "#aabbcc">
    < LinearLayout
        android:id = "@ + id/linearLayout1"
        android:layout_width = "match_parent"
```

```
            android:layout_height = "wrap_content" >
            <Button
                android:id = "@ + id/button1"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:text = "播放动画" />
            <Button
                android:id = "@ + id/button2"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:text = "停止动画" />
        </LinearLayout>
        <RadioGroup
              android:id = "@ + id/radioGroup1"
              android:layout_width = "wrap_content"
              android:layout_height = "wrap_content"
              android:orientation = "horizontal" >
        <RadioButton
            android:id = "@ + id/radioButton1"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:checked = "true"
            android:text = "单次播放" />
        <RadioButton
            android:id = "@ + id/radioButton2"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "循环播放" />
        </RadioGroup>
        <TextView
            android:id = "@ + id/textView1"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "拖动进度条修改透明度(0～255)之间" />
        <SeekBar
            android:id = "@ + id/seekBar1"
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content" />
        <ImageView
            android:id = "@ + id/imageView1"
            android:layout_width = "200dip"
            android:layout_height = "200dip"
            android:background = "@anim/frame" />
    </LinearLayout>
```

(4) 打开 src\fs.frameanimation_test 包下的 MainActivity.java 文件，在文件中实现动画的播放、停止，实现单次播放、循环播放及控件动画的透明度等功能。代码为：

```
package fs.frameanimation_test;
import android.os.Bundle;
import android.app.Activity;
```

```
import android.graphics.drawable.AnimationDrawable;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.RadioGroup.OnCheckedChangeListener;
import android.widget.SeekBar;
import android.widget.SeekBar.OnSeekBarChangeListener;
import android.view.*;
import android.view.View.OnClickListener;
public class MainActivity extends Activity {
    /** 第 1 次调用 activity 活动 */
    private Button button1,button2;
    private RadioGroup radioGroup;
    private RadioButton radioButton1,radioButton2;
    private SeekBar seekBar;
    private ImageView imageView1;
    private AnimationDrawable animationDrawable;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        button1 = (Button) this.findViewById(R.id.button1);
        button2 = (Button) this.findViewById(R.id.button2);
        radioGroup = (RadioGroup) this.findViewById(R.id.radioGroup1);
        radioButton1 = (RadioButton) this.findViewById(R.id.radioButton1);
        radioButton2 = (RadioButton) this.findViewById(R.id.radioButton2);
        seekBar = (SeekBar) this.findViewById(R.id.seekBar1);
        imageView1 = (ImageView) this.findViewById(R.id.imageView1);
        //通过 ImageView 对象拿到背景显示的 AnimationDrawable
        animationDrawable = (AnimationDrawable) imageView1.getBackground();
        button1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //TODO 自动存根法
                if(!animationDrawable.isRunning()){
                animationDrawable.start();
                }
            }
        });
        button2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //TODO 自动存根法
                if(animationDrawable.isRunning()){
                animationDrawable.stop();
                }
            }
        });
        radioGroup.setOnCheckedChangeListener(new OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(RadioGroup group,int checkedId) {
                //TODO 自动存根法
                if(checkedId == radioButton1.getId()){
```

```
                    //设置单次播放
                    animationDrawable.setOneShot(true);
                }else if(checkedId == radioButton2.getId()){
                    //设置循环播放
                    animationDrawable.setOneShot(false);
                }
                //设置播放后重新启动
                animationDrawable.stop();
                animationDrawable.start();
            }
        });
        //监听进度条修改的透明度
        seekBar.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {
            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
                //TODO 自动存根法
            }
            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
                //TODO 自动存根法
            }
            public void onProgressChanged(SeekBar seekBar, int progress,
                    boolean fromUser) {
                //TODO 自动存根法
                //设置动画 Alpha 值
                animationDrawable.setAlpha(progress);
                //通知 imageView 刷新屏幕
                imageView1.postInvalidate();
            }
        });
    }
}
```

运行程序，单击播放动画，选择“循环播放”按钮以及把拖动条拖放到最右边，效果如图 6-1(a)所示，当把拖动条拖放在到中间，效果如图 6-1(b)所示。




(a) 帧动画播放



(b) 改变动画的透明度

图 6-1　帧动画

6.2 Android 补间动画

View Animation(Tween Animation,补间动画)给出两个关键帧,通过一些算法将给定属性值在给定的时间内在两个关键帧间进行渐变。

View animation 只能应用于 View 对象,而且只支持一部分属性,如支持缩放旋转而不支持背景颜色的改变。而且对于 View animation,它只是改变了 View 对象绘制的位置,而没有改变 View 对象本身,例如,有一个 Button,坐标为(100,100),Width:200,Height:50,而有一个动画使其变为 Width:100,Height:100,会发现动画过程中触发按钮单击的区域仍是(100,100)~(300,150)。

View Animation 就是一系列 View 形状的变换,如大小的缩放、透明度的改变、位置的改变,动画的定义既可以用代码定义也可以用 XML 定义,当然,建议用 XML 定义。可以给一个 View 同时设置多个动画,例如从透明至不透明的淡入效果,与从小到大的放大效果,这些动画可以同时进行,也可以在一个完成之后再开始另一个。

用 XML 定义的动画放在/res/anim/文件夹内,XML 文件的根元素可以为<alpha>、<scale>、<translate>、<rotate>,interpolator 元素或<set>(表示以上几个动画的集合,set 可以嵌套)。在默认情况下,所有动画都是同时进行的,可以通过 startOffset 属性设置各个动画的开始偏移(开始时间)来达到动画顺序播放的效果。可以通过设置 interpolator 属性改变动画渐变的方式,例如 AccelerateInterpolator,开始时慢,然后逐渐加快。默认为 AccelerateDecelerateInterpolator。默认为 AccelerateDecelerateInterpolator。

Tween Animation 动画的 XML 使用格式为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android"
    android:interpolator = "@[package:]anim.interpolator_resource"
    android:shareInterpolator = ["true"|"false"]>
    < alpha android:fromAlpha = "float"
          android:toAlpha = "float"/>
    < scale android:fromXScale = "float"
          android:toXScale = "float"
          android:fromYScale = "float"
         android:toXScale = "float"
        android:pivotY = "float"/>
    < translate android:fromX = "float"
        android:toX = "float"
        android:fromY = "float"
        android:toY = "float"/>
    < rotate android:fromDegrees = "float"
          android:toDegrees = "float"
          android:pivotX = "float"
          android:pivotY = "float"/>
    < set >…
        </set >
</set >
```

从 XML 配置文件中可以看到，补间动画可以支持 alpha 淡入淡出、scale 缩放、translate 移动和 rotate 旋转等。

补间动画的通用方法主要有：

- setDuration(long durationMills)：设置动画持续时间(单位：毫秒)。
- setFillAfter(Boolean fillAfter)：如果 fillAfter 的值为 true，则动画执行后，控件将停留在执行结束的状态。
- setFillBefore(Boolean fillBefore)：如果 fillBefore 的值为 true，则动画执行后，控件将回到动画执行之前的状态。
- setStartOffSet(long startOffSet)：设置动画执行之前的等待时间。
- setRepeatCount(int repeatCount)：设置动画重复执行的次数。

6.2.1 Android 图像旋转

使用 Android 提供的 android.graphics.Matrix 类的 setRotate()、postRoate()和 preRotate()方法，可以对图像进行旋转。

由于这 3 个方法除了方法名不同外，语法格式等均相同，下面将以 setRotate()方法为例来进行介绍。setRoate()方法有以下同种语法格式。

- setRoate(float,degress)：使用该格式可以控件 Matrix 进行旋转，float 类型的参数用于指定旋转的角度。例如，创建一个 Matrix 的对象，并将其旋转 30°，可使用以下代码：

```
Matrix matrix = new Matrix();          //创建一个 Matrix 对象
matrix.setRoate(30);                   //将 Matrix 的对象旋转 30°
```

- setRotate(float degrees,float px,float py)：使用该格式可以控件 Matrix 以参数 px 和 py 为轴心进行旋转，float 类型的参数用于指定旋转的角度。例如，创建一个 Matrix 对象，并将其以(10,10)为轴心旋转 30°，可以使用以下代码：

```
Matrix matrix = new Matrix();          //创建一个 Matrix 的对象
matrix.setRoate(30,10,10);             //将 Matrix 的对象旋转 30°
```

创建 Matrix 的对象并对其进行旋转后，还需要应用该 Matrix 对图像或组件进行控件。在 Canvas 类中提供了一个 drawBitmap(Bitmap bitmap,Matrix matrix,Paint paint)方法，可以在绘制图像的同时应用 Matrix 上的变化。例如，需要将一个图像旋转 30°后绘制到画布上，可以使用以下代码：

```
Paint paint = new Paint();
Bitmap bitmap = BitmapFactory.decodeResource(MainActivity.this.getResources(),R.drawable.rabbit);
Matrix matrix = new Matrix();
matrix.setRotate(30);
canvas.drawBitmap(bitmap,matrix,paint);
```

下面通过一个实例来演示图像的旋转。

【例 6-2】 利用 rotate 控件实现动画的旋转。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Rotate_test。

(2) 打开 res\layout 目录下的 main. xml 布局文件，在文件中声明两个 Button 控件及一个 ImageView 控件。代码为：

```
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
    <Button
        android:id = "@ + id/bt1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignRight = "@ + id/button1"
        android:layout_below = "@ + id/button1"
        android:layout_marginTop = "26dp"
        android:text = "开始动画" />
    <Button
        android:id = "@ + id/bt2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_below = "@ + id/bt1"
        android:text = "取消动画" />
<ImageView
        android:id = "@ + id/imgView"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/bt1"
        android:layout_below = "@ + id/bt2"
        android:layout_marginTop = "67dp"
        android:src = "@drawable/big" />
</LinearLayout>
```

(3) 打开 src\fs. rotate_test 包下的 MainActivity. java 文件，在文件中实现动画的旋转及动画的取消。代码为：

```
package fs.rotate_test;
import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Matrix;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.animation.Animation;
import android.view.animation.RotateAnimation;
import android.widget.Button;
import android.widget.ImageView;
public class MainActivity extends Activity {
    ImageView image;
    Button start;
```

```
        Button cancel;
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            image = (ImageView) findViewById(R.id.imgView);
            start = (Button) findViewById(R.id.bt1);
            cancel = (Button) findViewById(R.id.bt2);
            /** 设置旋转动画 */
            final RotateAnimation animation = new RotateAnimation(0f,360f,Animation.RELATIVE_TO
_SELF,0.5f,Animation.RELATIVE_TO_SELF,0.5f);
            animation.setDuration(3000);//设置动画持续时间
            /** 常用方法 */
            start.setOnClickListener(new OnClickListener() {
                public void onClick(View arg0) {
                    image.setAnimation(animation);
                    /** 开始动画 */
                    animation.startNow();
                    }
                });
            cancel.setOnClickListener(new OnClickListener() {
                public void onClick(View v) {
                    /** 结束动画 */
                    animation.cancel();
                }
        });
        }
}
```

运行程序，默认效果如图 6-2(a)所示。当单击屏幕中的“开始动画”按钮时，图像开始旋转，效果如图 6-2(b)所示，当单击屏幕中的“取消动画”按钮时，图像旋转停止。




(a) 默认效果



(b) 旋转效果

图 6-2　旋转图像

6.2.2 Android 图像缩放

使用 Android 提供的 android.graphics.Matrix 类的 setScale()、postScale()和 preScale()方法,可对图像进行缩放。由于这 3 个方法除了方法名不同外,语法格式均相同,下面将以 setScale()方法为例来进行介绍。setScale()方法有以下两种语法格式。

- setScale(float sx,float sy):使用该格式可以控件 Matrix 进行缩放,参数 sx 和 sy 用于指定 X 轴和 Y 轴的缩放比例。例如,创建一个 Matrix 对象,并将其在 X 轴上缩放 40%,在 Y 轴上缩放 30%,可使用以下代码:

```
Matrix matrix = new Matrix();        //创建一个 Matrix 对象
matrix.setScale(0.4f,0.3f);          //缩放 Matrix 对象
```

- setScale(float sx,float sy,float px,float py):使用该格式可以控件 Matrix 以参数 px 和 py 为轴心进行缩放,参数 sx 和 sy 用于指定 X 和 Y 轴的缩放比例。例如,创建一个 Matrix 的对象,并将其以(100,100)为轴心,在 X 轴和 Y 轴上均缩放 40%,可以使用以下代码:

```
Matrix matrix = new Matrix();        //创建一个 Matrix 的对象
matrix.setScale(40,40,100,100);      //缩放 Matrix 对象
```

创建 Matrix 的对象并对其进行缩放后,还需要应用该 Matrix 对图像或组件进行控件。同旋转图像一样,也可应用 Canvas 类中提供的 drawBitmap(Bitmap bitmap,Matrix matrix,Paint paint)方法,在绘制图像的同时应用 Matrix 上的变化。

Scale 中定义可实现缩放的关键属性有:

- fromXScale:起始时 x 坐标的尺寸,设置为 1.0 说明是整个图片 x 轴的长度。
- toXScale:结束时 x 坐标的尺寸,设置为 0.0 说明整个图片 x 轴完全收缩到无。
- fromYScale:起始时 y 坐标的尺寸,设置为 1.0 说明是整个图片 y 轴的长度。
- toYScale:结束时 y 坐标的尺寸,设置为 1.0 说明是在收缩时 y 轴的长度保持不变。
- pivotX:动画在 X 轴方向缩放的中心点,值取 0%~100%或 0%p~100%p,50%为相对于自己的中心位置,50%p 表示相对于父控件的中心位置,也就是 p(parent)的意义。
- pivotY:动画在 Y 轴方向缩放的中心点,值取 0%~100%或 0%p~100%p,50%为相对于自己的中心位置,50%p 表示相对于父控件的中心位置,也就是 p(parent)的意义。
- duration:是设置动画的执行时间。
- interpolator:指定一个动画的插入器。Interpolator 定义一个动画的变化率,这使得基本的动画效果(alpha、scale、translate、rotate)得以加速、减速、重复等。Android 提供了几个 Interpolator 子类,实现了不同的速度曲线。
  - ◇ linear_interpolator:使动画以均匀的速率改变。
  - ◇ cycle_interolator:动画循环播放特定的次数,速率改变沿着正弦曲线。
  - ◇ accelerate_decelerate_interpolator:在动画开始与结束的地方速率改变比较慢,在中间时加速。

◇ accelerate_interpolator：在动画开始的地方速率改变比较慢，然后开始加速。

◇ decelerate_interpolator：在动画开始的地方改变比较慢，然后开始减速。

- zAdjustment：定义动画的 Z 轴方向的位置，值 normal 保持位置不变，top 保持在最上层，bottom 保持在最下层。
- repeatCount：动画的重复次数。
- repeatMode：定义重复的模式，其中值 restart 表示重新开始，reverse 表示先倒退再执行，倒退也算一次。

下面通过一个实例利用图像的缩放实现翻牌效果。

【例 6-3】 在 Android 中实现图像的缩放。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Scale_test。

(2) 打开 res\layout 目录下的 main. xml 文件，在文件中声明一个 ImageView 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = " # aabbcc">
< ImageView
    android:id = "@ + id/imgView"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_marginTop = "200px"
    android:src = "@drawable/pu2"/>
</LinearLayout >
```

(3) 打开 src\fs. scale_test 包下的 MainActivity. java 文件，在文件中实现当单击图像时实现图像的翻转。代码为：

```
package fs.com.scale_test;
import android.os.Bundle;
import android.widget.ImageView;
import android.app.Activity;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
public class MainActivity extends Activity {
    /** 第 1 次调用 activity 活动 */
    private ImageView imgView;
    //声明一个 boolean 用来切换背面和正面
    private boolean bool  =  false;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        imgView  =  (ImageView) findViewById(R.id.imgView);
```

```
            //给 ImageView 添加单击事件
            imgView.setOnClickListener(new ImgViewListener());
        }
        class ImgViewListener implements OnClickListener {
            @Override
            public void onClick(View v) {
                //TODO 自动存根法
                /* 也可通过代码来实现,这个是收缩效果
                AnimationSet animation = new AnimationSet(true);
                ScaleAnimation scale = new ScaleAnimation(1,0f,1,1f,
                    Animation.RELATIVE_TO_SELF,0.5f,
                    Animation.RELATIVE_TO_SELF,0.5f);
                animation.addAnimation(scale);
                animation.setDuration(150);
                 */
                //通过 AnimationUtils 得到动画配置文件(/res/anim/back_scale.xml)
     Animation animation = AnimationUtils.loadAnimation(MainActivity.this,R.anim.back);
                animation.setAnimationListener(new Animation.AnimationListener() {
                    @Override
                    public void onAnimationStart(Animation animation) {
                    }
                    @Override
                    public void onAnimationRepeat(Animation animation) {
                    }
                    @Override
                    public void onAnimationEnd(Animation animation) {
                        if(bool){
                            imgView.setImageResource(R.drawable.pu2);
                            bool = false;
                        }else {
                            imgView.setImageResource(R.drawable.puk1);
                            bool = true;
                        }
                        //通过 AnimationUtils 得到动画配置文件(/res/anim/front_scale.xml),然
                        //后在把动画交给 ImageView
                       imgView.startAnimation(AnimationUtils.loadAnimation(MainActivity.this,
R.anim.front));
                    }
                });
                imgView.startAnimation(animation);
            }
        }
    }
```

(4) 实现牌的翻转还需要实现两个动画的配置文件。在 res 中创建一个 anim 文件,在文件中新建两个文件,分别为 back.xml 及 front.xml。这两个文件的变化都是先对于某一点来变化的,因此,pivotX 和 pivotY 就是确定这个点的位置。在一个数轴上(原点为图片的左上角,x 轴和 y 轴的射线分别是向右和向下)。代码分别如下:

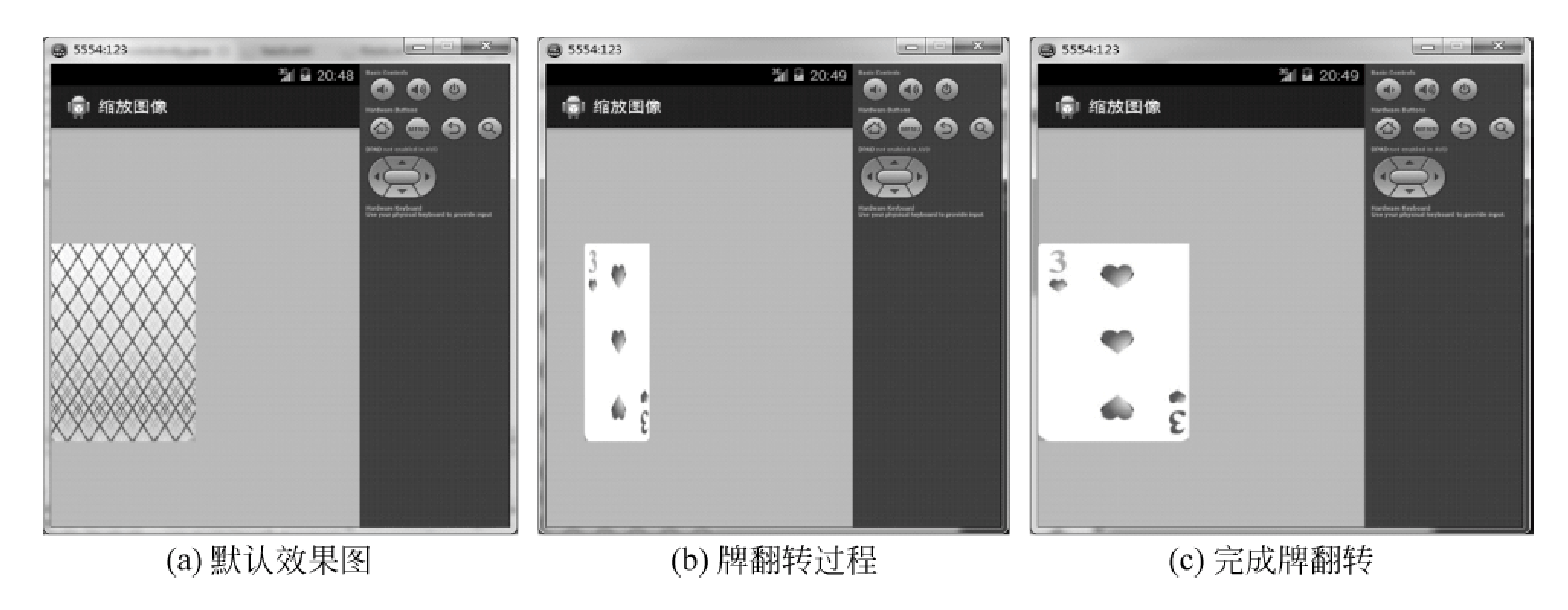
back.xml 文件的代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android"
      android:interpolator = "@android:anim/accelerate_interpolator">
      < scale
         android:fromXScale = "1.0"
         android:toXScale = "0.0"
         android:fromYScale = "1.0"
         android:toYScale = "1.0"
         android:pivotX = "50 % "
         android:pivotY = "50 % "
         android:duration = "150"/>
</set >
```

front.xml 文件的代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android"
      android:interpolator = "@android:anim/accelerate_interpolator">
      < scale
         android:fromXScale = "0.0"
         android:toXScale = "1.0"
         android:fromYScale = "1.0"
         android:toYScale = "1.0"
         android:pivotX = "50 % "
         android:pivotY = "50 % "
         android:duration = "150"/>
</set >
```

运行程序，默认效果如图 6-3(a)所示，当单击图中的图片，图像进行缩放，实现了牌的翻转，效果如图 6-3(b)及图 6-3(c)所示。



(a) 默认效果图　　(b) 牌翻转过程　　(c) 完成牌翻转

图 6-3　图像缩放

6.2.3　Android 倾斜图像

使用 Android 提供的 android.graphics.Matrix 类的 setSkew()、postSkew() 和 preSkew()方法，可对图像进行倾斜。由于这 3 个方法除了方法名不同外，语法格式均相同，下面将以 setSkew()方法为例来进行介绍。setSkew()方法有以下两种格式。

- setSkew(float kx,float ky)：使用该格式可以控件 Matrix 进行倾斜，参数 kx 和 ky 用于指定在 X 轴和 Y 轴上的倾斜量。例如，创建一个 Matrix 对象，并在 X 轴上倾斜 0.45，在 Y 轴上不倾斜，可以使用以下代码：

```
Matrix matrix = new Matrix();            //创建一个 Matrix 的对象
matrix.setSkew(0.45f,0);                 //倾斜 Matrix 对象
```

- setSkew(float kx,float ky,float px,float py)：使用该语法可以控件 Matrix 以参数 px 和 py 为轴心进行倾斜，参数 sx 和 sy 用于指定在 X 轴和 Y 轴上的倾斜量。例如，创建一个 Matrix 的对象，并将其以(100,100)为轴心，在 X 轴和 Y 轴上均倾斜 0.2，可使用以下代码：

```
Matrix matrix = new Matrix();            //创建一个 Matrix 的对象
matrix.setSkew(0.2f,0.2f,100,100);       //倾斜 Matrix 对象
```

创建 Matrix 的对象并对其进行倾斜后，还需要应用该 Matrix 对图像或组件进行控件。同旋转图像一样，也可以应用 Canvas 类中提供的 drawBitmap(Bitmap bitmap,Matrix matrix,Paint paint)方法，在绘制图像的同时应用 Matrix 上的变化。

下面通过一个实例来演示图像的倾斜。

【例 6-4】 在 Android 中实现图像的倾斜。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Skew_test。

(2) 打开 res\layout 目录下的 main.xml 文件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< FrameLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@ + id/frameLayout1"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = " # aabbcc">
</FrameLayout >
```

(3) 打开 src\fs.skew_test 包下的 MainActivity.java 文件，在文件中实现图像的倾斜。代码为：

```
package fs.skew_test;
import android.app.Activity;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Matrix;
import android.graphics.Paint;
import android.os.Bundle;
import android.view.View;
import android.widget.FrameLayout;
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
            setContentView(R.layout.main);
            //获取布局文件中的帧布局管理器
            FrameLayout ll = (FrameLayout)findViewById(R.id.frameLayout1);
            ll.addView(new MyView(this));                    //将自定义视图添加到帧布局管理器中
        }
        public class MyView extends View{
            public MyView(Context context) {
                super(context);
            }
            @Override
            protected void onDraw(Canvas canvas) {
                Paint paint = new Paint();                   //定义一个画笔
                paint.setAntiAlias(true);
                 Bitmap bitmap_bg = BitmapFactory.decodeResource(MainActivity.this.getResources(),
R.drawable.kp);
                canvas.drawBitmap(bitmap_bg,0,0,paint);   //绘制背景
                Bitmap bitmap_rabbit = BitmapFactory.decodeResource(MainActivity.this.getResources(),
R.drawable.big);
Matrix matrix = new Matrix();          //应用 setSkew(float sx,float sy)方法倾斜图像
matrix.setSkew(2f,1f);                 //以(0,0)点为轴心将图像在 X 轴上倾斜 2,在 Y 轴上倾斜 1
canvas.drawBitmap(bitmap_rabbit,matrix,paint);  //绘制图像并应用 matrix 的变换
                //应用 setSkew(float sx,float sy,float px,float py) 方法倾斜图像
                Matrix m = new Matrix();
                m.setSkew( - 0.5f,0f,78,69);   //以(78,69)点为轴心将图像在 X 轴上倾斜 - 0.5
            canvas.drawBitmap(bitmap_rabbit,m,paint);         //绘制图像并应用 matrix 的变换
                canvas.drawBitmap(bitmap_rabbit,0,0,paint);  //绘制原图
                super.onDraw(canvas);
            }
        }
}
```

运行程序,效果如图 6-4 所示。



图 6-4　图像倾斜

6.2.4 Android 图像平移

使用 Android 提供的 android. graphics. Matrix 类的 setTranslate()、postTranslate()和 preTranslate()方法，可对图像进行平移。由于这 3 个方法除了方法不同外，语法格式均相同，下面将以 setTranslate()方法为例来进行介绍。setTranslate()方法的格式为：

```
setTranslate(float dx,float dy)
```

在该语法中，参数 dx 和 dy 用于指定将 Matrix 移动到的位置的 x 和 y 坐标。

例如，创建一个 Matrix 的对象，并将其平移到(100,60)的位置，可使用以下的代码：

```
Matrix matrix = new Matrix();              //创建一个 Matrix 的对象
matrix.setTranslate(100,60);               //将对象平移到(100,60)的位置
```

创建 Matrix 的对象并对其进行平移后，还需要应用 Matrix 对图像或组件进行控件。同旋转图像一样，也可应用 Canvas 类中提供的 drawBitmap(Bitmap bitmap, Matrix matrix, Paint paint)方法，在绘制图像的同时应用 Matrix 上的变化。

Translate 中定义可实现平移的关键属性有：

- float fromXDelta：动画开始的点离当前 View X 坐标上的差值。
- float toXDelta：动画结束的点离当前 View X 坐标上的差值。
- float fromYDelta：动画开始的点离当前 View Y 坐标上的差值。
- float toYDelta：动画结束的点离当前 View Y 坐标上的差值。

下面通过一个实例来演示图像的平移。

【例 6-5】 在 Android 中利用图像的平移实现菜单的显示与隐藏。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Translate_test。

(2) 打开 res\layout 目录下的 main. xml 文件，在文件中定义一个 RelativeLayout 布局、两个 LinearLayout 布局，并分别在两个 LinearLayout 布局声明一个 TextView 控件及一个 Button 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
< LinearLayout
    android:id = "@ + id/menu"
    android:layout_height = "100px"
    android:layout_width = "fill_parent"
    android:layout_alignParentTop = "true"
    android:background = "#F3F3F3">
   < TextView
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
         android:text = "我是一个菜单"
         android:gravity = "center" />
</LinearLayout >
```

```
<Button
    android:id = "@ + id/button"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:layout_alignParentBottom = "true"
    android:text = "单击即显示或隐藏菜单" />
</RelativeLayout>
```

(3) 打开 src\fs.translate_test 包下的 MainActivity.java 文件，在文件中实现当单击界面中的按钮时，实现菜单的隐藏或显示。代码为：

```
package fs.translate_test;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.animation.Animation;
import android.view.animation.TranslateAnimation;
import android.widget.Button;
import android.widget.LinearLayout;
public class MainActivity extends Activity
{
    /** 第 1 次调用活动 */
    //Translate 动作的隐藏与显示
    Animation showAction,hideAction;
    LinearLayout menu;
    Button button;
    boolean menuShowed;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        menu = (LinearLayout) findViewById(R.id.menu);
        button = (Button)findViewById(R.id.button);
        //这里是 TranslateAnimation 动画
        showAction = new TranslateAnimation(
                Animation.RELATIVE_TO_SELF,0.0f,Animation.RELATIVE_TO_SELF,0.0f,
                Animation.RELATIVE_TO_SELF, -1.0f,Animation.RELATIVE_TO_SELF,0.0f);
        showAction.setDuration(500);
        //这里是 TranslateAnimation 动画
        hideAction = new TranslateAnimation(
                Animation.RELATIVE_TO_SELF,0.0f,Animation.RELATIVE_TO_SELF,0.0f,
                Animation.RELATIVE_TO_SELF,0.0f,Animation.RELATIVE_TO_SELF, -1.0f);
        hideAction.setDuration(500);
        menuShowed = false;
        menu.setVisibility(View.GONE);
        button.setOnClickListener(new OnClickListener()
        {
            @Override
            public void onClick(View v)
```

```
            {
                //TODO:自动存根法
                if (menuShowed)
                {
                    menuShowed = false;
                    menu.startAnimation(hideAction);
                    menu.setVisibility(View.GONE);
                }
    else
                {
                    menuShowed = true;
                    menu.startAnimation(showAction);
                    menu.setVisibility(View.VISIBLE);
                }
            }
        });
    }
}
```

运行程序，默认效果如图 6-5(a)所示，当单击界面中的按钮时，效果如图 6-5(b)所示。



(a) 默认界面



(b) 显示菜单

图 6-5　图像的平移

6.2.5　Android 透明度渐变

在 Android 中提供了 Alpha 对象用于实现图像的淡出淡入效果，主要控制的是透明度的变化。Alpha 中定义可实现淡出淡入效果的关键属性有：

- fromAlpha：表示动画起始时的透明度，0.0 表示完全透明，1.0 表示完全不透明，其取值在 0.0～1.0 之间的 float 数据类型的数字。
- toAlpha：表示动画结束时的透明度，其取值在 0.0～1.0 之间的 float 数据类型的数字。
- duration：动画持续时间，单位为毫秒。

下面用一个实例来演示图像的透明度渐变效果。

【例 6-6】 在 Android 中实现图像的透明度渐变效果。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 alpha_test。

(2) 打开 res\layout 目录下的 main.xml 文件，在文件中声明一个 ImageView 控件及一个 TextView 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = " # aabbcc">
< ImageView
    android:id = "@ + id/ImageView01"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"/>
< TextView
    android:id = "@ + id/TextView01"
    android:layout_below = "@ id/ImageView01"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "现在的 alpha 值是: " />
</LinearLayout >
```

(3) 打开 src\fs.alpha_test 包下的 MainActivity.java 文件，在文件中实现图像的透明度渐变效果，并把相应的 Alpha 值显示在 TextView 控件中。代码为：

```
package fs.alpha_test;
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.widget.ImageView;
import android.widget.TextView;
public class MainActivity extends Activity {
    //声明 ImageView 对象
    ImageView imageView;
    //声明 TextView
    TextView textView;
    //ImageView 的 alpha 值
    int image_alpha = 255;
    //Handler 对象用来给 UI_Thread 的 MessageQueue 发送消息
    Handler mHandler;
    //判断线程是否运行的变量
    boolean isrung = false;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        isrung = true;
```

```java
        //获得 ImageView 的对象
        imageView = (ImageView) this.findViewById(R.id.ImageView01);
        textView = (TextView) this.findViewById(R.id.TextView01);
        //设置 imageView 的图片资源.同样可以再 xml 布局中像下面这样写
        imageView.setImageResource(R.drawable.kp);
        //设置 imageView 的 Alpha 值
        imageView.setAlpha(image_alpha);
        //开启一个线程来让 Alpha 值递减
        new Thread(new Runnable() {
            @Override
            public void run() {
                while (isrung) {
                    try {
                        Thread.sleep(200);
                        //更新 Alpha 值
                        updateAlpha();
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        }).start();
        //接收消息之后更新 imageview 视图
        mHandler = new Handler() {
            @Override
            public void handleMessage(Message msg) {
                super.handleMessage(msg);
                imageView.setAlpha(image_alpha);
                //设置 textview 显示当前的 Alpha 值
                textView.setText("现在的 alpha 值是:" + Integer.toString(image_alpha));
                //刷新视图
                imageView.invalidate();
            }
        };
    }
    //更新 Alpha
    public void updateAlpha() {
        if (image_alpha - 7 >= 0) {
            image_alpha -= 7;
        } else {
            image_alpha = 0;
            isrung = false;
        }
        //发送需要更新 imageview 视图的消息 -->这里是发给主线程
        mHandler.sendMessage(mHandler.obtainMessage());
    }
}
```

运行程序,效果如图 6-6 所示。





图 6-6　图像透明度渐变效果

6.2.6　Android 补间动画的综合实例

在前面章节已经对补间动画的各种效果进行介绍，并给出相应的实例进行说明。下面通过一个综合实例来演示补间动画的综合应用。

【例 6-7】　实现移动补间动画、缩放补间动画、旋转补间动画、透明补间动画。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Tween_test。

(2) 在 res 下创建一个 anim 文件夹，并在文件夹中创建一个补间动画文件 animation.xml，用于实现 4 种补间动画。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:interpolator = "@android:anim/linear_interpolator">
<! -- 透明度变化  -->
     < alpha
         android:fromAlpha = "1"
         android:toAlpha = "0"
         android:duration = "2000"/>
<! -- 缩放与扩大  -->
     < scale android:fromXScale = "1.0"
        android:toXScale = "0"
        android:fromYScale = "1.0"
        android:toYScale = "0"
        android:pivotX = "50 % "
        android:pivotY = "50 % "
        android:fillAfter = "true"
        android:duration = "2000"/>
<! -- 水平与垂直位移  -->
    < translate
        android:fromXDelta = "0"
        android:toXDelta = "130"
```

```
            android:fromYDelta = "0"
            android:toYDelta = " - 80"
            android:duration = "2000"/>
    <! -- 旋转 -- >
        < rotate
            android:fromDegrees = "0"
            android:toDegrees = "360"
            android:pivotX = "50 % "
            android:pivotY = "50 % "
            android:duration = "2000"/>
    </set >
```

(3) 打开 res\layout 目录下的 main. xml 布局文件，在文件中声明一个 ImageView 控件及一个 Button 控件。代码为：

```
< RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc">
< ImageView
    android:id = "@ + id/image"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_alignParentLeft = "true"
    android:layout_alignParentTop = "true"
    android:layout_marginLeft = "84dp"
    android:layout_marginTop = "69dp"
    android:src = "@drawable/c1" />
< Button
    android:id = "@ + id/button"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_alignLeft = "@ + id/image"
    android:layout_below = "@ + id/image"
    android:layout_marginTop = "80dp"
    android:text = "动画" />
</RelativeLayout >
```

(4) 打开 src\fs. tween_test 包下的 MainActivity. java 文件，在文件中实现当单击按钮时，同时实现 4 种补间动画。代码为：

```
package com.example.tween_test;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.animation.Animation;
```

```
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.ImageView;
public class MainActivity extends Activity {
    private ImageView image;
    private Button button;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Animation anim = AnimationUtils.loadAnimation(this,R.anim.animation);
        button = (Button)findViewById(R.id.button);
        image = (ImageView)findViewById(R.id.image);
        button.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                //TODO 自动存根法
                image.startAnimation(anim);
            }
        });
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main,menu);
        return true;
    }
}
```

运行程序,效果如图 6-7 所示,单击界面中的“动画”按钮,可同时实现 4 种补间动画。



图 6-7　补间动画

6.2.7 Android 自定义补间动画

Android 提供了 Animation 作为补间动画抽象基类，而且为该抽象基类提供了 AlphaAnimation、RotateAnimation、ScaleAnimation、TranslateAnimation 4 个实现类，这 4 个实现类只是补间动画的 4 种基本形式：透明度改变、旋转、缩放、位移，在实际项目中可能还需要一些更复杂的动画，例如让图片在“三维”空间内进行旋转动画等，这就需要开发者自己开发补间动画了。

自定义补间动画并不难，需要继承 Animation，继承 Animation 时关键是要重写该抽象基类的 applyTransformation(float interpolatedTime,Transformation t)方法，该方法中两个参数的说明如下。

- interpolatedTime：代表了动画的时间进行比。不管动画实际的持续时间如何，当动画播放时，该参数总是自动从 0 变化到 1 的。
- Transformation：该参数代表了补间动画在不同时刻对图形或组件的变形程度。

从上面的介绍中可看出，实现自定义动画的关键在于重写 applyTransformation 方法时根据 interpolatedTime 时间来动态地计算动画对图片或视图的变形程度。

Transformation 代表了对图片或视图的变形，该对象里封装了一个 Matrix 对象，对它所包装的 Matrix 进行位移、倾斜、旋转等变换时，Transformation 将会控制对应的图片或视图进行相应的变换。

为了控制图片或 View 进行三维空间的变换，还需要借助于 Android 提供的一个 Camera，这个 Camera 并非代表手机的摄像头，只是一个空间变换工具，作用有点类似于 Matrix，但功能更强大。

Camera 提供了如下常用的方法。

- getMatrix(Matrix matrix)：将 Camera 所做的变换应用到指定 Matrix 上。
- rotateX(float deg)：将目标组件沿 *X* 轴旋转。
- rotateY(float deg)：将目标组件沿 *Y* 轴旋转。
- rotateZ(float deg)：将目标组件沿 *Z* 轴旋转。
- translate(float x,float y,float z)：把目标组件在三维空间进行位移变换。
- applyToCanvas(Canvas canvas)：把 Camera 所做的变换应用到 Canvas 上。

从上面的方法可看出，Camera 主要用于支持三维空间的变换，那么，手机中三维空间的坐标系统是怎样的呢？图 6-8 显示了手机屏幕上的三维坐标系统。

当 Camera 控制图片或 View 沿 *X*、*Y* 或 *Z* 轴旋转时，被旋转的图片或 View 将会呈现出三维透视的效果。图 6-9 显示了一张图片沿 *Y* 轴旋转的效果。

下面将通过一个实例来演示怎样实现自定义补间动画。

图 6-8　手机屏幕上的三维坐标

【例 6-8】 利用 Camera 来自定义在三维空间的动画。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 space_3d。

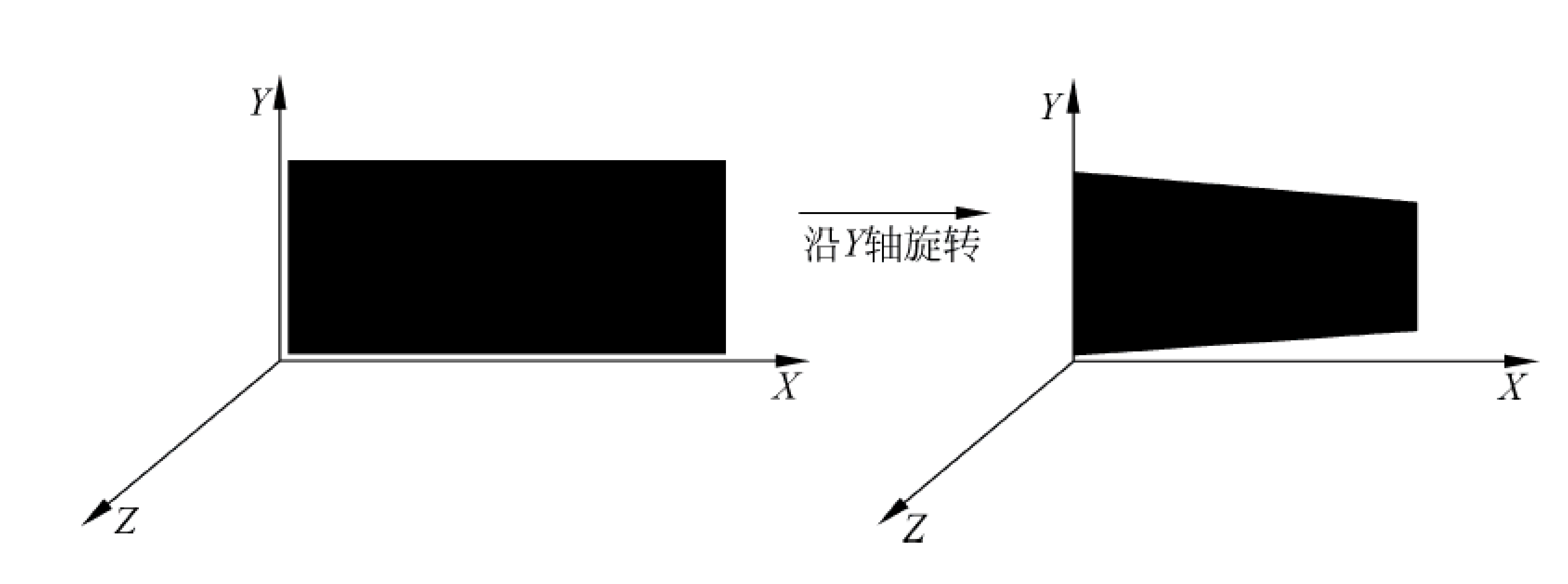


图 6-9　三维空间内沿 Y 轴旋转的效果

(2) 打开 res\layout 目录下的 main.xml 文件,在文件中声明一个 ListView 控件。代码为:

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc" >
<ListView
    android:id = "@+id/list"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:entries = "@array/cutomanimation"/>
</RelativeLayout>
```

(3) 在 value 文件夹中创建一个 array.xml 文件,用于存储 ListView 的内容。代码为:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<resources>
    <string-array name = "cutomanimation">
        <item>补间动画</item>
        <item>Android 动画的演示</item>
        <item>Android 补间动画    自定义补间动画</item>
        <item>义补间动画</item>
        <item>自定义补间动画</item>
    </string-array>
</resources>
```

(4) 打开 src\fs.space_3d 包下的 MainActivity.java 文件,在文件中实现根据动画的进行时间来控制 View 在 X 轴、Y 轴上的旋转,即实现三维空间内旋转效果。代码为:

```
package fs.space_3d;
import android.graphics.Camera;
import android.graphics.Matrix;
import android.view.animation.Animation;
```

```
import android.view.animation.LinearInterpolator;
import android.view.animation.Transformation;
public class MainActivity extends Animation
{
    private int centerX;
    private int centerY;
    //定义动画的持续事件
    private int duration;
    private Camera camera = new Camera();
    public MainActivity(int centerX,int centerY,int duration)
    {
        this.centerX = centerX;
        this.centerY = centerY;
        this.duration = duration;
    }
    @Override
    public void initialize(int width,int height,int parentWidth,
        int parentHeight)
    {
        super.initialize(width,height,parentWidth,parentHeight);
        //设置动画的持续时间
        setDuration(duration);
        //设置动画结束后的效果保留
        setFillAfter(true);
        setInterpolator(new LinearInterpolator());
    }
    /*
     * 该方法的 interpolatedTime 代表了抽象的动画持续时间,不管动画实际持续时间有多长
     * interpolatedTime 参数总是从 0(动画开始时)~1(动画结束时)
     * Transformation 参数代表了对目标组件所做的变化
     */
    @Override
    protected void applyTransformation(float interpolatedTime,Transformation t)
    {
        camera.save();
        //根据 interpolatedTime 时间来控制 X、Y、Z 上的偏移
        camera.translate(100.0f - 100.0f * interpolatedTime,
            150.0f * interpolatedTime - 150,
            80.0f - 80.0f * interpolatedTime);
        //根据 interpolatedTime 时间设置在 Y 柚上旋转不同的角度
        camera.rotateY(360 * (interpolatedTime));
        //根据 interpolatedTime 时间设置在 X 柚上旋转不同的角度
        camera.rotateX((360 * interpolatedTime));
        //获取 Transformation 参数的 Matrix 对象
        Matrix matrix = t.getMatrix();
        camera.getMatrix(matrix);
        matrix.preTranslate( - centerX, - centerY);
        matrix.postTranslate(centerX,centerY);
        camera.restore();
    }
}
```

(5) 在 src\fs. space_3d 包下创建一个 ListView. java 文件,该文件中既可以自定义动画来控制图片,也可以控制 View 组件。代码为:

```
package fs.space_3d;
import android.app.Activity;
import android.os.Bundle;
import android.view.Display;
import android.view.WindowManager;
import android.widget.ListView;
public class ListActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //获取 ListView 组件
        ListView list = (ListView)findViewById(R.id.list);
        WindowManager windowManager = (WindowManager)
            getSystemService(WINDOW_SERVICE);
        Display display = windowManager.getDefaultDisplay();
        //获取屏幕的宽和高
        int screenWidth = display.getWidth();
        int screenHeight = display.getHeight();
        //对 ListView 组件设置应用动画
        list.setAnimation(new MainActivity(screenWidth /2,screenHeight /2,3500));
    }
}
```

(6) 打开 AndroidManifest. xml 文件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.space_3d"
    android:versionCode = "1"
    android:versionName = "1.0" >
    < uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
    < application
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        < activity
            android:name = "fs.space_3d.ListActivity"
            android:label = "@string/app_name" >
            < intent - filter >
                < action android:name = "android.intent.action.MAIN" />
                < category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter >
```

```
        </activity>
    </application>
</manifest>
```

运行程序,ListView 控件的内容将会以三维变换的动画方式出现,效果如图 6-10 所示。



图 6-10　三维空间变换动画

6.3　Android 帧动画与补间动画综合实例

很多实际的动画往往同时运行两个,例如在此要实现一个小游戏,需要让用户控制游戏中的主角移动,当主角移动时,不仅要控制它的位置改变,还应该在它移动时播放 Frame 动画来让用户感觉更“逼真”。

下面为一个利用 Tween 动画与 Frame 动画开发的一个“老鹰飞翔”的效果,在这个实例中,老鹰飞翔时的振翅效果为 Frame 动画,而老鹰飞翔时的位置为 Tween 动画。其实现操作步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 EagleFly。

(2) 在 res 根目录下新建一个 anim 子目录文件下新建一个 laoyangfly. xml 文件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- 定义动画循环播放 -->
< animation - list xmlns:android = "http://schemas.android.com/apk/res/android"
    android:oneshot = "false">
    < item
        android:drawable = "@drawable/ly1" android:duration = "120" />
```

```
    < item
        android:drawable = "@drawable/ly2" android:duration = "120" />
    < item android:drawable = "@drawable/ly3" android:duration = "120" />
    < item
        android:drawable = "@drawable/ly4" android:duration = "120" />
    < item
        android:drawable = "@drawable/ly5" android:duration = "120" />
    < item
        android:drawable = "@drawable/ly6" android:duration = "120" />
</animation - list >
```

(3) 打开 res/Layout 目录下的 main. java 文件,在文件中声明一个 ImageView 控件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout
xmlns:apk = "http://schemas.android.com/apk/res/android"
apk:orientation = "vertical"
apk:layout_width = "fill_parent"
apk:layout_height = "fill_parent">
< ImageView
apk:id = "@ + id/laoyangfly"
apk:layout_width = "wrap_content"
apk:layout_height = "wrap_content"
apk:background = "@anim/laoyangfly" />
</LinearLayout >
```

(4) 打开 src/fs. eaglefly 包下的 MainActivity. java 文件,在文件中实现补间动画与帧动画。代码为:

```
package fs.eaglefly;
import java.util.Timer;
import java.util.TimerTask;
import android.app.Activity;
import android.graphics.drawable.AnimationDrawable;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.animation.TranslateAnimation;
import android.widget.ImageView;
public class MainActivity extends Activity
{
    //记录老鹰 ImageView 当前的位置
    private float curX = 0;
    private float curY = 30;
    //记录老鹰 ImageView 下一个位置的坐标
    private AnimationDrawable animDance;
    float nextX = 0;
    float nextY = 0;
```

```
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    //获取显示老鹰的 ImageView 组件
    final ImageView imageView = (ImageView)findViewById(R.id.laoyangfly);
    final Handler handler = new Handler()
    {
        @Override
        public void handleMessage(Message msg)
        {
            if (msg.what == 0x123)
            {
                //横向上一直向右飞
                if(nextX > 320)
                {
                    curX = nextX = 0;
                }
                else
                {
                    nextX += 8;
                }
                //纵向上可以随机上下
                nextY = curY + (float)(Math.random() * 10 - 5);
                //设置显示老鹰的 ImageView 发生位移改变
                TranslateAnimation anim
                    = new TranslateAnimation(curX,nextX,curY,nextY);
                curX = nextX;
                curY = nextY;
                anim.setDuration(200);
                //开始位移动画
                imageView.startAnimation(anim);
            }
        }
    };
    final AnimationDrawable butterfly = (AnimationDrawable)imageView
        .getBackground();
imageView.setOnClickListener(new OnClickListener()
    {
        @Override
        public void onClick(View v)
        {
            //开始播放老鹰振翅的逐帧动画
            butterfly.start();
            //通过定制器控制每 0.2 秒运行一次 TranslateAnimation 动画
            new Timer().schedule(new TimerTask()
            {
                @Override
                public void run()
                {
```

```
                        handler.sendEmptyMessage(0x123);
                    }
                },0,50);
            }
        });
    }
}
```

运行程序，单击图片，效果如图 6-11 所示。

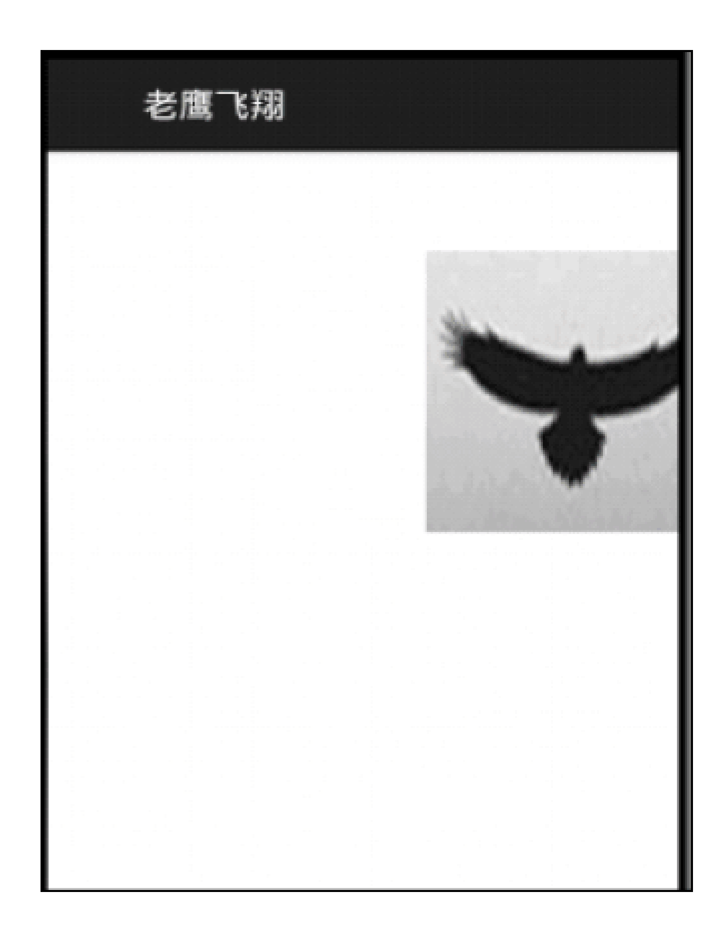


图 6-11 补间动画与帧动画实例

6.4 Android 动画渲染器

Android 动画技术中的 Interpolators 指的是动画在某一种形态下的速率，也称为动画渲染器，例如在旋转状态下使用某一个 Interpolators 对象来设置这个旋转的方式为加速或减速，或实现一些具有反弹效果的旋转样式，这时就要使用 Interpolators 对象了。

使用它的方式是在动画配置文件 XML 中的<set>标签中设置 android:interpolator 属性值即可。代码为：

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/accelerate_decelerate_interpolator">
```

总体来说，Interpolator 定义了动画变化的速率，提供不同的函数定义变化值相对于时间的变化规则，可以定义各种各样的非线性变换函数，例如加速、减速等。

Android 系统自带了多种 Interpolator 可以实现多种动画变化效果：

- AccelerateDecelerateInterpolator：先加速再减速；
- AccelerateInterpolator：加速；
- AnticipateInterpolator：先回退一小步，然后再迅速前进；
- AnticipateOvershootInterpolator：先回退一小步，然后再迅速前进，在超过右边界一小步；
- BounceInterpolator：实现弹球效果；

- CycleInterpolator：周期运动；
- DecelerateInterpolator：减速；
- LinearInterpolator：匀速；
- OvershootInterpolator：快速前进到右边界上，再往外突出一小步。

这些 Interpolator 可以应用于任意的 Animation 中，如 TranslateAnimation、RotateAnimation、ScaleAnimation、AlphaAnimation 等，其插值的对象随 Animation 种类不同而不同。例如对于 TranslateAnimation 来说，插值的对象是位移，对应的动画是平移的速率。对于 RotateAnimation 来说插值的对象为旋转角度，对应的动画为旋转的速率。

下面通过一个实例来演示 Interpolators 控件的使用，实现动画的渲染效果。

【例 6-9】 本实例用于使用 Spinner 控件选择对应动画效果，实现动画渲染。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Interpolators_test。

(2) 打开 res\layout 目录下 main.xml 文件，在文件中声明一个 ImageView 控件及一个 Spinner 控件。代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc">
    <ImageView
        android:id = "@ + id/target"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignParentBottom = "true"
        android:layout_marginBottom = "96dp"
        android:layout_marginLeft = "25dp"
        android:layout_toRightOf = "@ + id/target"
        android:src = "@drawable/ic_launcher" />
    <Spinner
        android:id = "@ + id/spinner1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/textView1"
        android:layout_below = "@ + id/textView1"
        android:layout_marginTop = "83dp" />
</RelativeLayout>
```

(3) 打开 src\fs.interpolators_test 包下的 MainActivity.java 文件，在文件中实现在 Spinner 控件中列出多个动画效果，当选择某个动画时，图像实现对应的渲染效果。代码为：

```
package fs.interpolators_test;
```

```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.view.animation.TranslateAnimation;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
public class MainActivity extends Activity implements
        AdapterView.OnItemSelectedListener {
    private static final String[] INTERPOLATORS = { "Accelerate","Decelerate",
            "Accelerate/Decelerate","Anticipate","Overshoot",
            "Anticipate/Overshoot","Bounce" };
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Spinner s = (Spinner) findViewById(R.id.spinner1);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_spinner_item, INTERPOLATORS);
        adapter
        .setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        s.setAdapter(adapter);
        s.setOnItemSelectedListener(this);
    }
    public void onItemSelected(AdapterView parent,View v,int position,long id) {
        final View target = findViewById(R.id.target);
        final View targetParent = (View) target.getParent();
        Animation animation = new TranslateAnimation(0.0f,targetParent
                .getWidth()
                - target.getWidth()
                - targetParent.getPaddingLeft()
                - targetParent.getPaddingRight(),0.0f,0.0f);
        animation.setDuration(1000);
        animation.setStartOffset(300);
        animation.setRepeatMode(Animation.RESTART);
        animation.setRepeatCount(Animation.INFINITE);
        switch (position) {
        case 0:
            animation.setInterpolator(AnimationUtils.loadInterpolator(this,
                    android.R.anim.accelerate_interpolator));
            break;
        case 1:
            animation.setInterpolator(AnimationUtils.loadInterpolator(this,
                    android.R.anim.decelerate_interpolator));
            break;
        case 2:
            animation.setInterpolator(AnimationUtils.loadInterpolator(this,
                    android.R.anim.accelerate_decelerate_interpolator));
            break;
```

```
            case 3:
                animation.setInterpolator(AnimationUtils.loadInterpolator(this,
                        android.R.anim.anticipate_interpolator));
                break;
            case 4:
                animation.setInterpolator(AnimationUtils.loadInterpolator(this,
                        android.R.anim.overshoot_interpolator));
                break;
            case 5:
                animation.setInterpolator(AnimationUtils.loadInterpolator(this,
                        android.R.anim.anticipate_overshoot_interpolator));
                break;
            case 6:
                animation.setInterpolator(AnimationUtils.loadInterpolator(this,
                        android.R.anim.bounce_interpolator));
                break;
            }
            target.startAnimation(animation);
        }
        public void onNothingSelected(AdapterView parent) {
        }
    }
```

运行程序,效果如图 6-12 所示。



图 6-12　动画的渲染效果

6.5 Android 动画组件

本节将介绍在 Android 中常用的动画组件(ViewAnimator),通过这些组件能方便地实现一组 View 动画。

动画组件的基类为 FrameLayout,作用是为 FrameLayout 里面的 View 切换提供动画效果。

一般不直接使用 ViewAnimator 而是使用它的子类 ViewFlipper 和 ViewSwitcher,其中 ViewSwitcher 的子类又包含了 ImageSwitcher 和 TextSwitcher。ViewFlipper 和 ViewSwitcher 的主要区别为 ViewSwitcher 最多能有两个子 View,而 ViewFlipper 可以有多个。

6.5.1 ViewSwitcher 组件

ImageSwitcher 是一个控制图片切换显示的组件,可添加图片切换动画,效果很好,很适合做相册,或动态展示图片。在 Android 中还有一个比较类似的组件就是 TextSwitcher,它们的用法基本相同。

使用 ImageSwitcher 或者 TextSwitcher 必须设置一个 ViewFactory,主要用来在 ViewSwitcher 中创建 View,因此需要实现 ViewSwitcher. ViewFactory 接口,最后通过 makeView()方法创建相应的 View,即 ImageSwitcher 对应 ImageView,TextSwitcher 对应 TextView。

下面通过一个实例来介绍 ViewFactory 的用法。

【例 6-10】 仿 Android 系统 Launcher 界面,实现分屏左右滑动效果。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 ViewFactory_test。

(2) 打开 res\layout 目录下的 main. xml 布局文件,在文件中声明一个 ViewSwitcher 控件及两个 Button 控件。代码为:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity"
    android:background="#aabbcc">
   <!-- 定义一个 ViewSwitcher 组件 -->
    <ViewSwitcher
        android:id="@+id/viewSwitcher"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />
    <!-- 定义滚动到上一屏的按钮 -->
```

```
    <Button
        android:id = "@ + id/button_prev"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignParentBottom = "true"
        android:layout_alignParentLeft = "true"
        android:onClick = "prev"
        android:text = "上一页"/>
    <! -- 定义滚动到下一屏的按钮 -->
    <Button
        android:id = "@ + id/button_next"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignParentBottom = "true"
        android:layout_alignParentRight = "true"
        android:onClick = "next"
        android:text = "下一页" />
</RelativeLayout>
```

(3) 在 res\layout 目录下创建一个 labelicon.xml 文件,在文件中声明一个 ImageView 控件及一个 TextView 控件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- 定义一个垂直的 LinearLayout,该容器中放置一个 ImageView 和一个 TextView -->
<LinearLayout
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:gravity = "center"
    android:background = "#ccffee">
    <ImageView
        android:id = "@ + id/imageview"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"/>
    <TextView
        android:id = "@ + id/textview"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:gravity = "center"/>
</LinearLayout>
```

(4) 在 res\layout 目录下创建一个 slidelistview.xml 文件,在文件中声明一个 GridView 控件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<GridView
 xmlns:android = "http://schemas.android.com/apk/res/android"
 android:layout_width = "match_parent"
 android:numColumns = "4"
 android:layout_height = "match_parent">
```

```
</GridView>
```

(5) 在 res 文件夹下创建一个 anim 文件夹,在文件夹中分别创建两个名为 slide_in_right. xml、slide_out_left. xml 的文件,分别实现动画的左右拖动。

slide_in_right. xml 文件代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android">
    <! -- 设置从右边拖进来的动画
    android:duration 指定动画持续时间   -->
    < translate
        android:fromXDelta = "100 % p"
        android:toXDelta = "0"
        android:duration = "@android:integer/config_mediumAnimTime" />
</set>
```

slide_out_left. xml 文件代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android">
    <! -- 设置从左边拖出去的动画
    android:duration 指定动画持续时间  -->
    < translate
        android:fromXDelta = "0"
        android:toXDelta = " - 100 % p"
        android:duration = "@android:integer/config_mediumAnimTime" />
</set>
```

(6) 打开 src\fs. viewfactory_test 包下的 MainActivity. java 文件,在文件中实现屏幕的左右分屏效果。代码为:

```
package fs.viewfactory_test;
import java.util.ArrayList;
import android.os.Bundle;
import android.app.Activity;
import android.graphics.drawable.Drawable;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.ViewSwitcher;
import android.widget.ViewSwitcher.ViewFactory;
public class MainActivity extends Activity
{
    //定义一个常量,用于显示每屏显示的应用程序数
    public static final int NUMBER_PER_SCREEN  =  12;
    //代表应用程序的内部类
    public static class DataItem
```

```
{
    //应用程序名称
    public String dataName;
    //应用程序图标
    public Drawable drawable;
}
//保存系统所有应用程序的 List 集合
private ArrayList<DataItem> items = new ArrayList<DataItem>();
//记录当前正在显示第几屏的程序
private int screenNo = -1;
//保存程序所占用的总屏数
private int screenCount;
ViewSwitcher switcher;
//创建 LayoutInflater 对象
LayoutInflater inflater;
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    inflater = LayoutInflater.from(MainActivity.this);
    //创建一个包含 40 个元素的 List 集合,用于模拟包含 40 个的应用程序
    for (int i = 0; i < 40; i++)
    {
        String label = "" + i;
        Drawable drawable = getResources().getDrawable(
                R.drawable.ic_launcher);
        DataItem item = new DataItem();
        item.dataName = label;
        item.drawable = drawable;
        items.add(item);
    }
    //计算应用程序所占用的总屏数
//如果应用程序的数量能整除 NUMBER_PER_SCREEN,除法的结果就是总屏数
    //如果不能整除,总屏数应该是除法的结果再加 1
    screenCount = items.size() % NUMBER_PER_SCREEN == 0 ?
            items.size()/ NUMBER_PER_SCREEN :
            items.size() / NUMBER_PER_SCREEN + 1;
    switcher = (ViewSwitcher) findViewById(R.id.viewSwitcher);
    switcher.setFactory(new ViewFactory()
    {
        //实际上就是返回一个 GridView 组件
        @Override
        public View makeView()
        {
            //加载 R.layout.slidelistview 组件,实际上就是一个 GridView 组件
            return inflater.inflate(R.layout.slidelistview,null);
        }
    });
    //页面加载时先显示第一屏
    next(null);
```

```
    }
    public void next(View v)
    {
        if (screenNo < screenCount - 1)
        {
            screenNo++;
            //为 ViewSwitcher 的组件显示过程设置动画
            switcher.setInAnimation(this,R.anim.slide_in_right);
            //为 ViewSwitcher 的组件隐藏过程设置动画
            switcher.setOutAnimation(this,R.anim.slide_out_left);
            //控制下一屏将要显示的 GridView 对应的 Adapter
            ((GridView) switcher.getNextView()).setAdapter(adapter);
            //单击右边按钮,显示下一屏,也可通过手势检测实现显示下一屏
            switcher.showNext();
        }
    }
    public void prev(View v)
    {
        if (screenNo > 0)
        {
            screenNo--;
            //为 ViewSwitcher 的组件显示过程设置动画
            switcher.setInAnimation(this,android.R.anim.slide_in_left);
            //为 ViewSwitcher 的组件隐藏过程设置动画
            switcher.setOutAnimation(this,android.R.anim.slide_out_right);
            //控制下一屏将要显示的 GridView 对应的 Adapter
            ((GridView) switcher.getNextView()).setAdapter(adapter);
            //单击左边按钮,显示上一屏,也可通过手势检测实现显示上一屏
            switcher.showPrevious();
        }
    }
     //该 BaseAdapter 负责为每屏显示的 GridView 提供列表项
    private BaseAdapter adapter = new BaseAdapter()
    {
        @Override
        public int getCount()
        {
            //如果已经到了最后一屏,且应用程序的数量不能整除 NUMBER_PER_SCREEN
            if (screenNo == screenCount - 1
                    && items.size() % NUMBER_PER_SCREEN != 0)
            {
                //最后一屏显示的程序数为应用程序的数量对 NUMBER_PER_SCREEN 求余
                return items.size() % NUMBER_PER_SCREEN;
            }
            //否则每屏显示的程序数量为 NUMBER_PER_SCREEN
            return NUMBER_PER_SCREEN;
        }
        @Override
        public DataItem getItem(int position)
        {
            //根据 screenNo 计算第 position 个列表项的数据
```

```
            return items.get(screenNo * NUMBER_PER_SCREEN + position);
        }
        @Override
        public long getItemId(int position)
        {
            return position;
        }
        @Override
        public View getView(int position,
                View convertView,ViewGroup parent)
        {
            View view = convertView;
            if (convertView == null)
            {
                //加载 R.layout.labelicon 布局文件
                view = inflater.inflate(R.layout.labelicon,null);
            }
           //获取 R.layout.labelicon 布局文件中的 ImageView 组件,并为之设置图标
            ImageView imageView = (ImageView)
                    view.findViewById(R.id.imageview);
            imageView.setImageDrawable(getItem(position).drawable);
            //获取 R.layout.labelicon 布局文件中的 TextView 组件,并为之设置文本
            TextView textView = (TextView)
                    view.findViewById(R.id.textview);
            textView.setText(getItem(position).dataName);
            return view;
        }
    };
}
```

运行程序,效果如图 6-13(a)所示效果,单击界面中“下一页”或“上一页”按钮,即进行分屏,效果如图 6-13(b)所示。



(a) 第一屏　　(b) 第三屏

图 6-13　左右分屏效果

6.5.2 ViewFlipper 组件

屏幕切换(ViewFlipper)指的是在同一个 Activity 内屏幕可见的切换,最常见的情况就是在一个 FrameLayout 内有多个页面,例如一个系统设置页面;一个个性化设置页面。

通过查看 OPhone API 文档可以发现,android. widget. ViewAnimator 类继承自 FrameLayout,ViewAnimator 类的作用是为 FrameLayout 里面的 View 切换提供动画效果。该类有如下几个和动画相关的函数:

- setInAnimation:设置 View 进入屏幕时所使用的动画,该函数有两个版本,一个接收单个参数,类型为 android. view. animation. Animation;一个接收两个参数,类型为 Context 和 int,分别为 Context 对象和定义 Animation 的 resourceID。
- setOutAnimation:设置 View 退出屏幕的时候使用的动画,参数 setInAnimation 函数一样。
- showNext:调用该函数来显示 FrameLayout 里面的下一个 View。
- showPrevious:调用该函数来显示 FrameLayout 里面的上一个 View。

在 Android 中,一般不直接使用 ViewAnimator 而是使用它的两个子类 ViewFlipper 和 ViewSwitcher。ViewFlipper 可以用来指定 FrameLayout 内多个 View 之间的切换效果,可以一次指定也可以在每次切换的时候都指定单独的效果。该类额外提供了下面几个函数:

isFlipping:用来判断 View 切换是否正在进行。

setFilpInterval:设置 View 之间切换的时间间隔。

startFlipping:使用上面设置的时间间隔来开始切换所有的 View,切换会循环进行。

stopFlipping:停止 View 切换。

下面通过一个实例来演示 ViewFlipper 组件的用法。

【例 6-11】 实现动画的触摸。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 ViewFlipper_test。

(2) 打开 res\layout 目录下的 main. xml 文件,在文件中声明一个 ViewFlipper 组件,并内置两个子布局文件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "#aabbcc" >
    < ViewFlipper
        android:id = "@ + id/viewFlipper"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent" >
        < include
            android:id = "@ + id/layout1"
            layout = "@layout/layout1" />
        < include
            android:id = "@ + id/layout2"
            layout = "@layout/layout2" />
```

```
    </ViewFlipper>
</LinearLayout>
```

(3)在 res\layout 目录下创建一个 layhout1. xml 文件,用于第 1 个 View 界面,声明一个 TextView 控件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "#001122">
    <TextView
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:gravity = "center"
        android:text = "一个 TextView"
        android:textSize = "40dip" />
</LinearLayout>
```

(4) 在 res\layout 目录下创建一个 layhout2. xml 文件,用于第 2 个 View 界面,声明一个 TextView 控件及一个 ImageView 控件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "#aaffbb">
    <LinearLayout
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:gravity = "center"
        android:orientation = "vertical" >
        <ImageView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:src = "@drawable/ic_launcher" />
        <TextView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "一个 TextView + 一个 ImageView"
            android:textSize = "20dip" />
    </LinearLayout>
</LinearLayout>
```

(5) 在 res 下新建一个名为 anim 的文件夹,在文件夹中创建两个左右位移与透明度动画,分别命名为 slide_in_right. xml、slide_out_left. xml。代码分别如下。

```
slide_in_right.xml 文件代码为:
<?xml version = "1.0" encoding = "utf - 8"?>
```

```
<set xmlns:android = "http://schemas.android.com/apk/res/android">
    <translate android:fromXDelta = "50 % p"
        android:toXDelta = "0"
        android:duration = "300"/>
    <alpha android:fromAlpha = "0.0"
        android:toAlpha = "1.0"
        android:duration = "300" />
</set>
```

slide_out_left.xml 文件代码为：

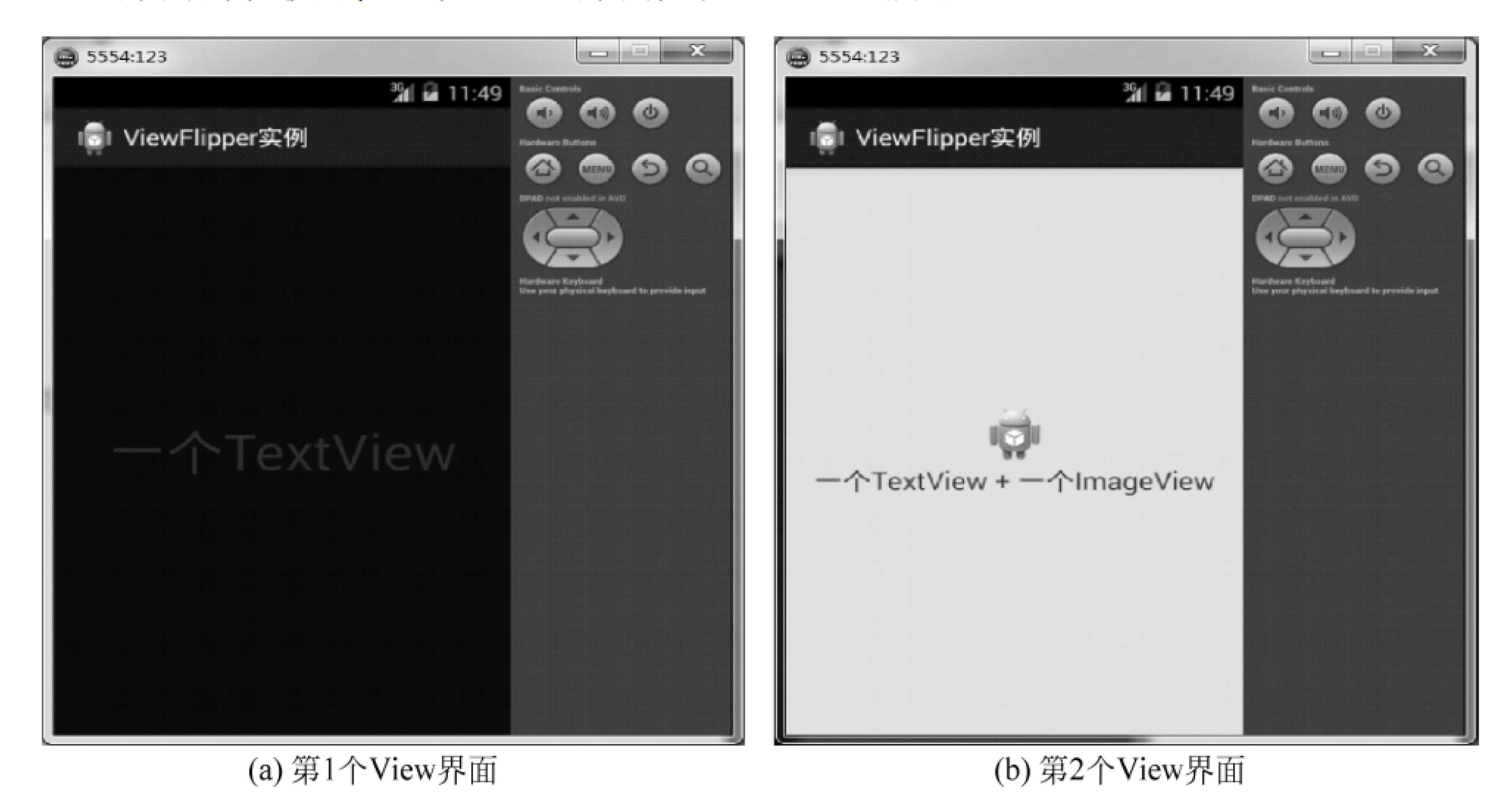
```
<?xml version = "1.0" encoding = "utf - 8"?>
<set xmlns:android = "http://schemas.android.com/apk/res/android">
    <translate android:fromXDelta = "0"
        android:toXDelta = " - 50 % p"
        android:duration = "300"/>
    <alpha android:fromAlpha = "1.0"
        android:toAlpha = "0.0"
        android:duration = "300" />
</set>
```

(6) 打开 src\fs.viewflipper_test 包下的 MainActivity.java 文件，用于实现两个 View 界面间的切换。代码为：

```
package fs.viewflipper_test;
import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.view.animation.AnimationUtils;
import android.widget.ViewFlipper;
public class MainActivity extends Activity implements OnTouchListener {
    private ViewFlipper viewFlipper;
    //左右滑动时手指按下的 X 坐标
    private float touchDownX;
    //左右滑动时手指松开的 X 坐标
    private float touchUpX;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        viewFlipper = (ViewFlipper) findViewById(R.id.viewFlipper);
        viewFlipper.setOnTouchListener(this);
    }
    @Override
    public boolean onTouch(View v,MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            //获得左右滑动时手指按下的 X 坐标
            touchDownX = event.getX();
            return true;
```

```
            } else if (event.getAction() == MotionEvent.ACTION_UP) {
                //获得左右滑动时手指松开的 X 坐标
                touchUpX = event.getX();
                //从左往右,看前一个 View
                if (touchUpX - touchDownX > 100) {
                    //设置 View 切换的动画
                    viewFlipper.setInAnimation(AnimationUtils.loadAnimation(this,
                            android.R.anim.slide_in_left));
                    viewFlipper.setOutAnimation(AnimationUtils.loadAnimation(this,
                            android.R.anim.slide_out_right));
                    //显示下一个 View
                    viewFlipper.showPrevious();
                    //从右往左,看后一个 View
                } else if (touchDownX - touchUpX > 100) {
                    //设置 View 切换的动画
                    //由于 Android 没有提供 slide_out_left 和 slide_in_right,所以仿照 slide_in_
left 和 slide_out_right 编写了 slide_out_left 和 slide_in_right
                    viewFlipper.setInAnimation(AnimationUtils.loadAnimation(this,R.anim.slide_
in_right));
                    viewFlipper.setOutAnimation(AnimationUtils.loadAnimation(this, R.anim.slide_
out_left));
                    //显示前一个 View
                    viewFlipper.showNext();
                }
                return true;
            }
            return false;
        }
}
```

运行程序,默认为第 1 个 View 界面,效果如图 6-14(a)所示,当左右拖动鼠标时即可实现 View 界面的切换,第 2 个 View 界面如图 6-14(b)所示。



(a) 第1个View界面　　(b) 第2个View界面

图 6-14　两个界面间的切换

6.6 SurfaceView 实现动画

在 View 中可实现绘图，但其绘图机制存在如下两个缺陷：

- View 缺乏双缓冲机制。
- 当程序需要更新 View 上的图像时，程序必须重绘 View 上显示的整张图片。

由于 View 存在上面两个缺陷，所以通过自定义 View 来实现绘图，尤其是在游戏中的绘图时，其性能并不好。Android 提供了一个 SurfaceView 来代替 View，在实现游戏绘图方面，SurfaceView 比 View 更加出色，因此，一般推荐使用 SurfaceView。

6.6.1 SurfaceView 绘制机制

SurfaceView 一般会与 SurfaceHolder 结合使用，SurfaceHolder 用于向与之关联的 SurfaceView 上绘图，调用 SurfaceView 的 getHolder()方法即可获取 SurfaceView 关联的 SurfaceHolder。

SurfaceHolder 提供了如下方法来获取 Canvas 对象。

- Canvas lockCanvas()：锁定整个 SurfaceView 对象，获取该 Surface 上的 Canvas。
- Canvas lockCanvas(Rect dirty)：锁定 SurfaceView 上 Rect 划分的区域，获取该 Surface 上的 Canvas。

当对同一个 SurfaceView 调用上面两个方法时，两个方法所返回的是同一个 Canvas 对象。但当程序调用第 2 个方法获取指定区域的 Canvas 时，SurfaceView 将只对 Rect 所“圈”出来的区域进行更新，通过这种方式可以提高画面的更新速度。

当通过 lockCanvas()获取指定了 SurfaceView 上的 Canvas 之后，接着程序就可以调用 Canvas 进行绘图了，Canvas 绘图完成后通过如下方法来释放绘图，提交所绘制的图形的代码为：

```
unlockCanvasAndPost(canvas);
```

需要指出的是，当调用 SurfaceHolder 的 unlockCanvasAndPost 方法后，该方法之前所绘制的图形还处于缓冲之中，下一次 lockCanvas()方法锁定的区域可能会“遮挡”它。

下面通过一个实例来演示 SurfaceView 的绘图机制。

【例 6-12】 利用 SurfaceView 绘制几个方块。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 SurfaceView_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明一个 SurfaceView 控件。代码为：

```
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:orientation = "vertical">
    <SurfaceView
        android:id = "@ + id/main_sv"
        android:layout_width = "match_parent"
```

```
        android:layout_height = "match_parent"/>
</LinearLayout>
```

(3) 打开 src\fs. surfaceview_test 包下的 MainActivity. java 文件，在文件中实现 SurfaceView 绘制正方形。其代码为：

```
package fs.surfaceview_test;
import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Rect;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceHolder.Callback;
import android.view.SurfaceView;
import android.view.View;
import android.view.View.OnTouchListener;
import fs.surfaceview_test.*;
public class MainActivity extends Activity {
    //SurfaceHolder 负责维护 SurfaceView 上绘制的内容
    private SurfaceHolder holder;
    private Paint paint;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        paint = new Paint();
        //获取 SurfaceView 实例
        SurfaceView surface = (SurfaceView) findViewById(R.id.main_sv);
        //初始化 SurfaceHolder 对象
        holder = surface.getHolder();
        holder.addCallback(new Callback() {
            //当 surface 将要被销毁时回调该方法
            @Override
            public void surfaceDestroyed(SurfaceHolder holder) {
            }
            //当 surface 被创建时回调该方法
            @Override
            public void surfaceCreated(SurfaceHolder holder) {
                //锁定整个 SurfaceView
                Canvas canvas = holder.lockCanvas();
                //获取背景资源
                Bitmap bitmap = BitmapFactory.decodeResource(
                        MainActivity.this.getResources(),
                        R.drawable.bj);
                //绘制背景
                canvas.drawBitmap(bitmap,0,0,null);
```

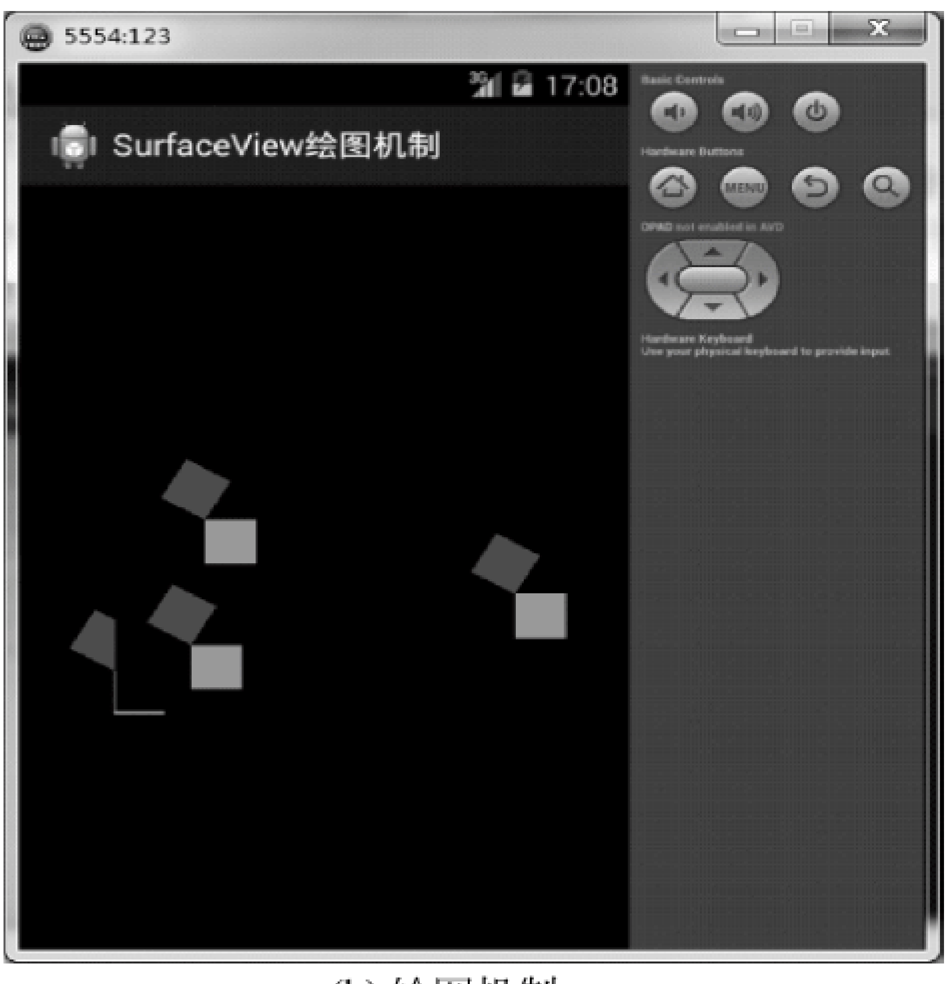
```
                //绘制完成,释放画布,提交修改
                holder.unlockCanvasAndPost(canvas);
                //重新锁两次,避免下次 lockCanvas 被遮挡
                holder.lockCanvas(new Rect(0,0,0,0));
                holder.unlockCanvasAndPost(canvas);
                holder.lockCanvas(new Rect(0,0,0,0));
                holder.unlockCanvasAndPost(canvas);
            }
            //当一个 surface 的格式或大小发生改变时回调该方法
            @Override
            public void surfaceChanged(SurfaceHolder holder,int format,
                    int width,int height) {
            }
        });
        surface.setOnTouchListener(new OnTouchListener() {
            @Override
            public boolean onTouch(View v,MotionEvent event) {
                //只处理按下事件
                if (event.getAction() == MotionEvent.ACTION_DOWN) {
                    int cx = (int) event.getX();
                    int cy = (int) event.getY();
                    //锁定 SurfaceView 的局部区域,只更新局部内容
                    Canvas canvas = holder.lockCanvas(new Rect(cx - 60,
                            cy - 60,cx + 60,cy + 60));
                    //保存 canvas 的当前状态
                    canvas.save();
                    //旋转画布
                    canvas.rotate(30,cx,cy);
                    paint.setColor(Color.RED);
                    //绘制红色方块
                    canvas.drawRect(cx - 40,cy - 40,cx,cy,paint);
                    //恢复 canvas 之前的保存状态
                    canvas.restore();
                    paint.setColor(Color.GREEN);
                    //绘制绿色方块
                    canvas.drawRect(cx,cy,cx + 40,cy + 40,paint);
                    //绘制完成,释放画布,提交修改
                    holder.unlockCanvasAndPost(canvas);
                }
                return false;
            }
        });
    }
}
```

上面的程序重写了 Callback 对象的 surfaceCreated 方法,并在该方法中为 SurfaceView 绘制了一个背景。为了避免背景图片被下一次 lockCanvas 遮挡,程序先调用了 holder.lockCanvas(new Rect(0,0,0,0));,本次 lockCanvas 会“遮挡”上次 lockCanvas 所绘制的图形,但由于本次 lockCanvas 的区域为 new Rect(0,0,0,0),因此这里绘制的背景以后就不会被遮挡了。

上面的程序监听了触摸屏事件，每次触碰屏幕时，程序会锁定触碰周围的区域(只更新该区域的数据)，而且本次 lockCanvas 会遮挡上次 lockCanvas 后绘制的图形。运行程序，默认效果如图 6-15(a)所示。SurfaceView 上的“遮挡”有点类似于 Flash 上“蒙版”的概念，例如图 6-15(b)所示的第 2 次绘图“遮挡”了第 1 次绘图；第 3 次 lockCanvas 时又可能“遮挡”第 2 次 lockCanvas 的区域，但不可能“遮挡”第 1 次 lockCanvas 的区域；如果第 2 次 lockCanvas“遮挡”的区域又被第 3 次 lockCanvas 所“遮挡”，那么原来第 1 次 drawCanvas 所绘制的图形可能会“显露”出来。



(a) 默认界面



(b) 绘图机制

图 6-15　SurfaceView 绘制正方形

6.6.2　利用 SurfaceView 开发示波器

SurfaceView 与普通 View 还有一个重要的区别：View 的绘图必须在当前 UI 线程中进行——这也是前面程序需要更新 View 组件的总要采用 Handler 处理的原因；但 SurfaceView 就不会存在这个问题，因为 SurfaceView 的绘图是由 SurfaceHolder 来完成的。

对于 View 组件，如果程序需要花较长的时间来更新绘图，那么主 UI 线程将会被阻塞，无法响应用户的任何动作；而 SurfaceViewHolder 则会启用新的线程去更新 SurfaceView 的绘制，因此不会阻塞主 UI 线程。

一般来说，如果程序或游戏界面的动画元素较多，而且很多都需要通过定时器来控制这些动画元素的移动，那么就可以考虑使用 SurfaceView，而不是 View。

下面通过利用 SurfaceView 控件来开发一个示波器程序，该程序将会根据用户单击的按钮在屏幕上自动绘制正弦波或余弦波。

【例 6-13】 利用 SurfaceView 控件绘制一个示波器。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 SurfaceView_Scope。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明两个 Button 控件及一个 SurfaceView 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
```

```
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
<LinearLayout android:orientation = "horizontal"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:gravity = "center">
<Button android:id = "@ + id/sin"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "正弦波"/>
<Button android:id = "@ + id/cos"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "余弦波"/>
</LinearLayout>
<SurfaceView android:id = "@ + id/show"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:gravity = "center"/>
</LinearLayout>
```

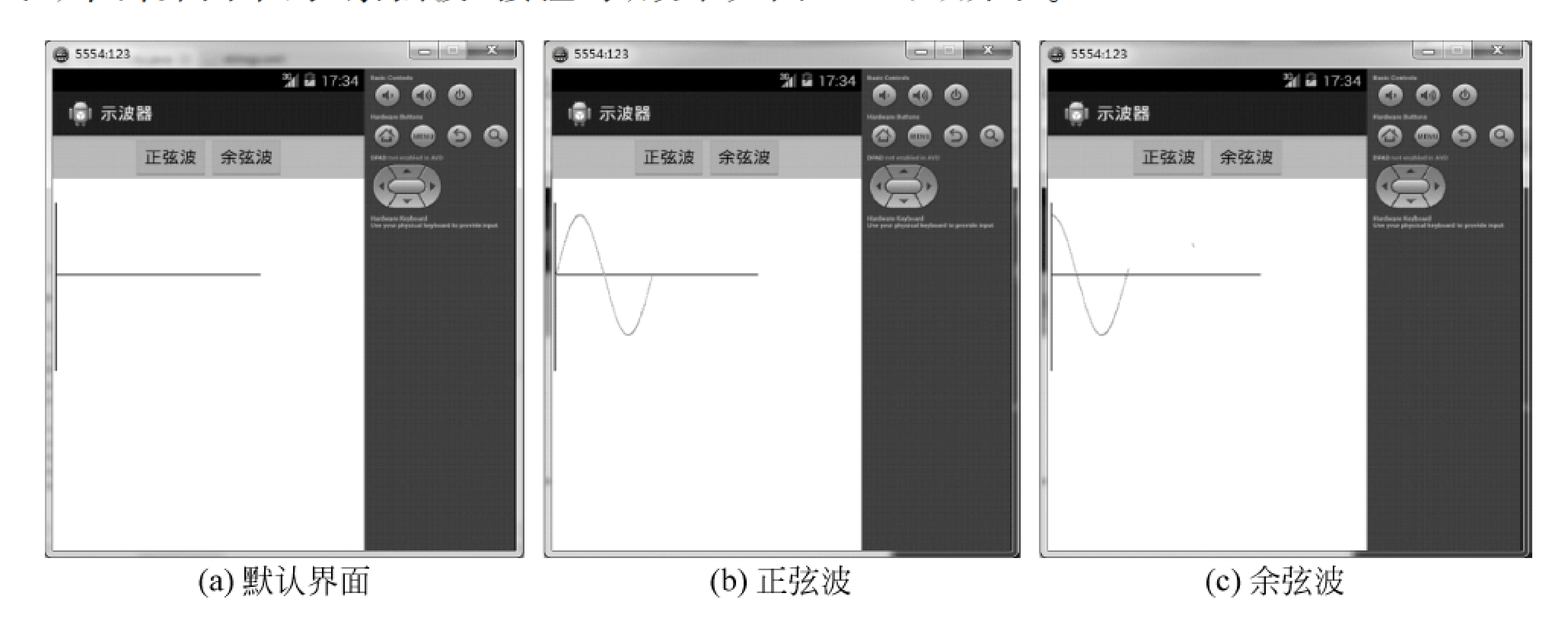
(3) 打开 src/fs. surfaceview_scope 包下的 MainActivity. java 文件,在文件中实现当单击界面中的"正弦波"按钮时,在坐标系中绘制正弦波,当单击界面中的"余弦波"按钮时,在坐标系中绘制余弦波。代码为:

```
package fs.surfaceview_scope;
import java.util.Timer;
import java.util.TimerTask;
import android.app.Activity;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Rect;
import android.os.Bundle;
import android.view.SurfaceHolder;
import android.view.SurfaceHolder.Callback;
import android.view.SurfaceView;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
public class MainActivity extends Activity
{
    private SurfaceHolder holder;
    private Paint paint;
    final int HEIGHT = 320;
    final int WIDTH = 320;
    final int X_OFFSET = 5;
    private int cx = X_OFFSET;
```

```
//实际的 Y 轴的位置
int centerY = HEIGHT / 2;
Timer timer = new Timer();
TimerTask task = null;
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    final SurfaceView surface = (SurfaceView) findViewById(R.id.show);
    //初始化 SurfaceHolder 对象
    holder = surface.getHolder();
    paint = new Paint();
    paint.setColor(Color.GREEN);
    paint.setStrokeWidth(3);
    Button sin = (Button)findViewById(R.id.sin);
    Button cos = (Button)findViewById(R.id.cos);
    OnClickListener listener = (new OnClickListener()
    {
        @Override
        public void onClick(final View source)
        {
            drawBack(holder);
            cx = X_OFFSET;
            if(task != null)
            {
                task.cancel();
            }
            task = new TimerTask()
            {
                public void run()
                {
                    int cy = source.getId() == R.id.sin ? centerY - (int)(100 *
                    Math.sin((cx - 5) * 2 * Math.PI / 150))
                    : centerY - (int)(100 * Math.cos((cx - 5) * 2 * Math.PI / 150));
        Canvas canvas = holder.lockCanvas(new Rect(cx,cy - 2,cx + 2  ,cy + 2));
                    canvas.drawPoint(cx,cy,paint);
                    cx ++;
                    if (cx > WIDTH)
                    {
                        task.cancel();
                        task = null;
                    }
                    holder.unlockCanvasAndPost(canvas);
                }
            };
            timer.schedule(task,0,30);
        }
    });
    sin.setOnClickListener(listener);
    cos.setOnClickListener(listener);
```

```
            holder.addCallback(new Callback()
            {
                @Override
                public void surfaceChanged(SurfaceHolder holder, int format,
                    int width, int height)
                {
                    drawBack(holder);
                }
                @Override
                public void surfaceCreated(final SurfaceHolder myHolder)
                {
                }
                @Override
                public void surfaceDestroyed(SurfaceHolder holder)
                {
                    timer.cancel();
                }
            });
        }
        private void drawBack(SurfaceHolder holder)
        {
            Canvas canvas = holder.lockCanvas();
            //绘制白色背景
            canvas.drawColor(Color.WHITE);
            Paint p = new Paint();
            p.setColor(Color.BLACK);
            p.setStrokeWidth(2);
            //绘制坐标轴
            canvas.drawLine(X_OFFSET,centerY,WIDTH,centerY,p);
            canvas.drawLine(X_OFFSET,40,X_OFFSET,HEIGHT,p);
            holder.unlockCanvasAndPost(canvas);
            holder.lockCanvas(new Rect(0,0,0,0));
            holder.unlockCanvasAndPost(canvas);
        }
    }
```

运行程序,默认效果如图 6-16(a)所示,单击界面中的“正弦波”按钮时,效果如图 6-16(b)所示,单击界面中的“余弦波”按钮时,效果如图 6-16(c)所示。



(a) 默认界面　　(b) 正弦波　　(c) 余弦波

图 6-16　SurfaceView 开发示波器

6.7 Android 图像扭曲

在 Android 中提供了 drawBitmapMesh 类实现了图像的扭曲 Canvas 的 drawBitmapMesh，定义如下：

```
public void drawBitmapMesh(Bitmap bitmap, int meshWidth, int meshHeight, float[] verts, int
vertOffset,int[] colors,int colorOffset,Paint paint)
```

其用于表示将图像绘制在网格上，简言之，可以将画板想象成一张格子布，在这张布上绘制图像。对于一个网格端点均匀分布的网格来说，横向有 meshWidth+1 个顶点，纵向有 meshHeight+1 个端点。顶点数组 verts 是以行优先的数组(二维数组以一维数组表示，先行后列)。网格可以不均匀分布，参数定义如下：

- Bitmap：需要绘制在网格上的图像。
- meshWidth：网格的宽度方向的数目(列数)，为 0 时不绘制图像。
- meshHeight：网格的高度方向的数目(行数)，为 0 时不绘制图像。
- verts：为(x，y)对的数组，表示网格顶点的坐标，至少需要有(meshWidth+1) * (meshHeight+1) * 2+meshOffset 个(x，y)坐标。
- vertOffset：用于控制 verts 数组中开始跳过的(x，y)对的数目。
- Colors：可以为空，不为空为每个顶点定义对应的颜色值，至少需要有(meshWidth+1) * (meshHeight+1) * 2+meshOffset 个(x，y)坐标。
- colorOffset：colors 数组中开始跳过的(x，y)对的数目。
- paint：可以为空。

值得注意的是，当程序希望调用 drawBitmapMesh 方法对位图进行扭曲时，关键是计算 verts 数组的值——该数组的值记录了扭曲后的位图上各"顶点"的坐标。

下面用实例来演示 drawBitmapMesh 控件的用法。

【例 6-14】 利用 drawBitmapMesh 控件对载入的图像进行扭曲处理。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 drawBitmapMesh_test。

(2) res\layout 目录下的 main.xml 布局文件采用默认代码。

(3) 打开在 src\fs.drawbitmapmesh_test 包下的 MainActivity.java 文件，在文件中实现单击图像的某项处理时，实现对应的扭曲处理，单击空白处时，图像还原回原始图像。代码为：

```
package fs.drawbitmapmesh_test;
import android.app.Activity;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.os.Bundle;
import android.util.AttributeSet;
import android.view.MotionEvent;
```

```
import android.view.View;
public class MainActivity extends Activity {
    /** 第 1 次调用 activity 活动 */
    private Bitmap bitmap;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new MyView(this,R.drawable.fj));
    }
    private class MyView extends View
    {
        //定义两个常量,这两个常量指定该图片横向、纵向上都被划分为 20 格
        private final int WIDTH = 20;
        private final int HEIGHT = 20;
        //记录该图片上包含的 441 个顶点
        private final int COUNT = (WIDTH + 1) * (HEIGHT + 1);
        //定义一个数组,记录 Bitmap 上的 21 * 21 个点的坐标
        private final float[] verts = new float[COUNT * 2];
        //定义一个数组,记录 Bitmap 上的 21 * 21 个点经过扭曲后的坐标
        //对图片扭曲的关键就是修改该数组里元素的值
        private final float[] orig = new float[COUNT * 2];
        public MyView(Context context,int drawableId) {
            super(context);
            setFocusable(true);
            //根据指定资源加载图片
            bitmap = BitmapFactory.decodeResource(getResources(),drawableId);
            //获取图片宽度和高度
            float bitmapWidth = bitmap.getWidth();
            float bitmapHeight = bitmap.getHeight();
            int index = 0;
            for(int y = 0; y <= HEIGHT; y++)
            {
                float fy = bitmapHeight * y / HEIGHT;
                for(int x = 0 ; x <= WIDTH; x++)
                {
                    float fx = bitmapWidth * x / WIDTH;
                    //初始化 orig、verts 数组
                  //初始化,orig、verts 两个数组均匀地保存了 21 * 21 个点的(x,y)坐标
                    orig[index * 2 + 0] = verts[index * 2 + 0] = fx;
                    orig[index * 2 + 1] = verts[index * 2 + 1] = fy;
                    index += 1;
                }
            }
            //设置背景色
            setBackgroundColor(Color.WHITE);
        }
        protected void onDraw(Canvas canvas)
        {
            //对 bitmap 按 verts 数组进行扭曲
            //从第 1 个点(由第 5 个参数 0 控制)开始扭曲
            canvas.drawBitmapMesh(bitmap,WIDTH,HEIGHT,verts,0,null,0,null);
        }
        //工具方法,用于根据触摸事件的位置计算 verts 数组里各元素的值
        private void warp(float cx,float cy)
```

```
        {
            for(int i = 0; i < COUNT * 2; i += 2)
            {
                float dx = cx - orig[i + 0];
                float dy = cy - orig[i + 1];
                float dd = dx * dx + dy * dy;
                //计算每个坐标点与当前点(cx,cy)之间的距离
                float d = (float)Math.sqrt(dd);
                //计算扭曲度,距离当前点(cx,cy)越远,扭曲度越小
                float pull = 80000 / ((float)(dd * d));
              //对 verts 数组(保存 bitmap 上 21 * 21 个点经过扭曲后的坐标)重新赋值
                if(pull >= 1)
                {
                    verts[i + 0] = cx;
                    verts[i + 1] = cy;
                }
                else
                {
                    //控制各顶点向触摸事件发生的点偏移
                    verts[i + 0] = orig[i + 0] + dx * pull;
                    verts[i + 1] = orig[i + 1] + dx * pull;
                }
            }
            //通知 View 组件重绘
            invalidate();
        }
        public boolean onTouchEvent(MotionEvent event)
        {
            //调用 warp 方法根据触摸屏事件的坐标点来扭曲 verts 数组
            warp(event.getX(),event.getY());
            return true;
        }
    }
}
```

运行程序,默认效果如图 6-17(a)所示,单击图像的某一处时图像实现扭曲处理,效果如图 6-17(b)及图 6-17(c)所示。



(a) 原始图像　　(b) 图像扭曲1　　(c) 图像扭曲2

图 6-17　图像的扭曲处理

第 7 章　Android 绘图

除了使用已有的图片之外，Android 应用常常需要在运行时动态地生成图片，例如一个手机游戏，游戏界面看上去丰富多彩，而且可以随着用户动作而动态改变，这就需要借助 Android 的绘图支持了。

7.1　Android 常用绘图

在 Android 中，绘制图像时最常应用的就是 Paint 类、Canvas 类、Bitmap 类和 BitmapFactory 类。其中 Paint 类代表画笔，Canvas 类代表画布。在现实生活中，有画笔和画布就可以正常作画了，在 Android 中也是如此，通过 Paint 类和 Canvas 类即可绘制图像。

7.1.1　Paint 类

Paint 类代表画笔，用来描述图形的颜色和风格，如线宽、颜色、透明度和填充效果等信息。使用 Paint 类时，需要先创建该类的对象，这可以通过该类提供的构造方法来实现。通常情况下，只需要使用 Paint()方法来创建一个使用默认设置的 Paint 对象。代码为：

```
Paint paint = new Paint();
```

创建 Paint 类的对象后，还可以通过该对象提供的方法来对画笔的默认设置进行改变，如改变画笔的颜色、笔触宽度等。用于改变画笔设置的常用方法如表 7-1 所示。

表 7-1　Paint 类的常用方法

方　　法	描　　述
setRGB(int a,int r,int g,int b)	用于设置颜色，各参数值均为 0～255 之间的整数，分别用于表示透明度、红色、绿色和蓝色值
setColor(int color)	用于设置颜色，参数 color 可以通过 Color 类提供的颜色常量指定，也可以通过 Color. rgb(int red,int green,int blue)方法指定
setAlpha(int a)	用于设置透明度，值为 0～255 之间的整数
setAntiAlias(boolean aa)	用于指定是否使用抗锯齿功能，如果使用会使绘图速度变慢
setDither(boolean dither)	用于指定是否使用图像抖动处理，如果使用会使图像颜色更加平滑和饱满，更加清晰
setPathEffect(PathEffect effect)	用于设置绘制路径时的路径效果，如点划线
setShader(Shader shader)	用于设置渐变，可以使用 LinearGradient(线性渐变)、RadialGradient(径向渐变)或者 SweepGradient(角度渐变)

续表

方　　法	描　　述
setShadowLayer(float radius, float dx, float dy, int color)	用于设置阴影，参数 radius 为阴影的角度，dx 和 dy 为阴影在 X 轴和 Y 轴上的距离，color 为阴影颜色。如果参数 radius 的值为 0，那么将没有阴影
setStrokeCap(Paint.Cap cap)	用于当画笔的填充样式为 STROKE 或 FILL_AND_STROKE 时，设置笔刷的图形样式，参数值可以是 Cap.BUTT、Cap.ROUND 或 Cap.SQUARE。主要体现在线的端点上
setStrokeJoin(Paint.Join join)	用于设置画笔转弯处的连接风格，参数值为 Join.BEVEL、Join.MITER 或 Join.ROUND
setStrokeWidth(float width)	用于设置笔触的宽度
setStyle(Paint.Style style)	用于设置填充风格，参数值为 Style.FILL、Style.FILL_AND_STROKE 或 Style.STROKE
setTextAlign(Paint.Align align)	用于设置绘制文本时的文字对齐方式，参数值为 Align.CENTER、Align.LEFT 或 Align.RIGHT
setTextSize(float textSize)	用于设置绘制文本时文字的大小
setFakeBoldText (boolean fakeBoldText)	用于设置是否为粗体文字
setXfermode(Xfermode xfermode)	用于设置图形重叠时的处理方式，如合并、取交集或并集，经常用来制作橡皮的擦除效果

如果要定义一个画笔，指定该画笔的颜色为绿色，带一个浅灰色的阴影，可以使用下面的代码：

```
Paint paint = new Paint();
paint.setColor(Color.green);
paint.setShadowLayer(2,3,3,Color.rgb(180,180,180));
```

下面通过一个实例来演示 Paint 类的用法。

【例 7-1】 使用 graphics 类来绘制了一幅简单的北京奥运宣传画，包括奥运五环，"北京欢迎您"的宣传标语以及福娃。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Paint_Test。

(2) res\layout 目录下的布局文件 main.xml 采用默认代码。

(3) 在 src\paint_canvas_test 包下新建一个自定义类 Custom_View，在类中重写了 onDraw()函数，并定义几种不同的画笔，分别用来绘制各种颜色的奥运五环以及绘制字符串"北京欢迎您"等。代码为：

```
package com.paint_test;
import android.view.View;
import android.content.Context;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Paint.Style;
public class Custom_View extends View {
```

```
    public Custom_View(Context context) {
        super(context);
        }
    public void onDraw(Canvas canvas) {
        Paint paint_blue = new Paint();               //绘制蓝色的环
        paint_blue.setColor(Color.BLUE);
        paint_blue.setStyle(Style.STROKE);
        paint_blue.setStrokeWidth(10);
        canvas.drawCircle(110,150,60,paint_blue);
        Paint paint_yellow = new Paint();             //绘制黄色的环
        paint_yellow.setColor(Color.YELLOW);
        paint_yellow.setStyle(Style.STROKE);
        paint_yellow.setStrokeWidth(10);
        canvas.drawCircle((float)175.5,210,60,paint_yellow);
        Paint paint_black = new Paint();              //绘制黑色的环
        paint_black.setColor(Color.BLACK);
        paint_black.setStyle(Style.STROKE);
        paint_black.setStrokeWidth(10);
        canvas.drawCircle(245,150,60,paint_black);
        Paint paint_green = new Paint();              //绘制绿色的环
        paint_green.setColor(Color.GREEN);
        paint_green.setStyle(Style.STROKE);
        paint_green.setStrokeWidth(10);
        canvas.drawCircle(311,210,60,paint_green);
        Paint paint_red = new Paint();                //绘制红色的环
        paint_red.setColor(Color.RED);
        paint_red.setStyle(Style.STROKE);
        paint_red.setStrokeWidth(10);
        canvas.drawCircle(380,150,60,paint_red);
        Paint paint_string = new Paint();             //绘制字符串
        paint_string.setColor(Color.BLUE);
        paint_string.setTextSize(20);
        canvas.drawText("Welcome to Beijing",245,310,paint_string);
        Paint paint_line = new Paint();               //绘制直线
        paint_line.setColor(Color.BLUE);
        canvas.drawLine(240,310,425,310,paint_line);
        Paint paint_text = new Paint();               //绘制字符串
        paint_text.setColor(Color.BLUE);
        paint_text.setTextSize(20);
        canvas.drawText("北京欢迎您",275,330,paint_text);
        //绘制福娃图片
        canvas.drawBitmap(BitmapFactory.decodeResource(getResources(),R.drawable.fuwa),35,
    340,paint_line);
        }
    }
```

(4) 打开 src\paint_test 包下的 MainActivity 文件，在文件中实现将自定义的 MyView 视图显示到手机屏幕上，即加载 MyView 视图，可以使用 setContentView()方法。代码为：

```
    package com.paint_test;
    import android.os.Bundle;
```

```
import android.app.Activity;
import android.view.Menu;
import android.view.MenuItem;
import android.support.v4.app.NavUtils;
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(newCustom_View(this));   //加载 MyView
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main,menu);
      return true;
    }
}
```

运行程序,效果如图 7-1 所示。



图 7-1　绘制 2D 图形效果

7.1.2　Canvas 类

Canvas 类代表画布,通过该类提供的方法,可以绘制各种图形(如矩形、圆形和线条等)。通常情况下,要在 Android 中绘图,需要创建一个继承自 View 类的视图,并且在该类中重写它的 onDraw(Canvas canvas)方法,然后在显示绘图的 Activity 中添加该视图。

Canvas 类提供的主要方法如表 7-2 所示。

表 7-2　Canvas 类的常用方法

方　　法	描　　述
drawColor(int color)	用于设置画布的背景颜色,可以通过 Color 类中的预定义颜色来设置,也可以通过指定 RGB 值来设置
drawLine(float startX,float startY, float stopX, float, stopY, Paint paint)	用于在画布上绘制直线,通过指定直线的两个端点坐标来绘制。该方法只能绘制单条直线;如果需要同时绘制多条直线,则可以使用 drawLines 方法。参数 startX 为起始端点的 X 坐标;参数 startY 为起始端点的 Y 坐标;参数 stopX 为终止端点的 X 坐标;stopY 参数为终止端点的 Y 坐标;参数 paint 为绘制直线所使用的画笔
drawLines (float [ ] pts, Paint paint)	用于在画布上绘制多条直线,通过指定直线的端点坐标数组来绘制。参数 pts 为绘制直线的端点数组,每条直线占用 4 个数据;参数 paint 为绘制直线所使用的画笔
drawPoint(float x,float y,Paint paint)	用于在画布上绘制一个点,通过指定端点坐标来绘制。该方法只能绘制单个点,如果需要同时绘制多个点,则可以使用 drawPoints 方法。参数 x 为绘制点的 X 坐标;参数 y 为绘制点的 Y 坐标;参数 paint 为绘制点所使用的画笔
drawPoints (float [ ] pts, int offset,int count,Paint paint)	用于在画布上绘制多个点,通过指定端点坐标数组来绘制。参数 pts 为绘制点的数组,每个端点占用两个数据;参数 offset 为跳过的数据个数,这些数据将不参与绘制过程;参数 count 为实际参与绘制的数据个数;参数 paint 为绘制时所使用的画笔
drawRect (float left, float top, float right, float below, Paint paint)	用于在画布上绘制矩形,可以通过指定矩形的 4 条边来实现,也可以通过指定 Rect 对象来实现。同时也可以通过设置画笔的空心效果来绘制空心的矩形。参数 r 为 Rect 对象;参数 rect 为 RectF 对象;参数 left 为矩形的左边位置;参数 top 为矩形的上边位置;参数 right 为矩形的右边位置;参数 below 为矩形的下边位置;参数 paint 为绘制矩形时所使用的画笔
drawRoundRect (RectF rect, float rx,float ry,Paint paint)	用于在画面上绘制圆角矩形,通过指定 RectF 对象及圆角半径来实现。参数 rect 为 RectF 对象;参数 rx 为 X 方向上的圆角半径;参数 ry 为 Y 方向上的圆角半径;参数 paint 为绘制时所使用的画笔
drawCircle (float cx, float cy, float radius,Paint paint)	用于在画布上绘制圆形,通过指定圆形圆心的坐标和半径来实现,该方法是绘制圆形的主要方法,同时也可以通过设置画笔的空心效果来绘制空心的圆形。参数 cx 的圆心 x 坐标;参数 cy 的圆心 y 坐标;参数 radius 为圆的半径;参数 paint 为绘制时所使用的画笔
drawOval(RectF oval,Paint paint)	用于在画布上绘制椭圆形,通过指定椭圆外切矩形的 RectF 对象来实现。该方法为绘制椭圆形的主要方法,同时也可以通过设置画笔的空心效果来绘制空心的椭圆形
drawPath(Path path,Paint paint)	用于在画布上绘制任意多边形,通过指定 Path 对象来实现。在 Path 对象中规划了多边形的路径信息
drawArc (RectF oval, float startAngle, float sweepAngle, Boolean useCenter,Paint paint)	用于在画布上绘制圆弧,通过指定圆弧所在的椭圆对象、起始角度、终止角度来实现

续表

方　法	描　述
drawText(String text, int start, int end, float x, float y, Paint paint)	用于在画布上绘制字符串,通过指定字符串的内容和显示的位置来实现。在画布上绘制字符串是经常性的操作,Android 系统提供了非常灵活的绘制字符串的方法,可以根据不同的需要调用不同的方法来实现。字体的大小、样式等信息都需要在 Paint 画笔中来指定
drawBitmap (Bitmap bitmap, float left, float top, Paint paint)	用于在画布上绘制图像,通过指定 Bitmap 对象来实现。前面的各个方法都是自己绘制各个图形,但应用程序往往需要直接引用一些图片资源。这时可使用 drawBitmap 方法来在画布上直接显示图像
save()	用于锁定画布,这种方法主要用于锁定画布中的某一个或几个对象,对锁定对象操作的场合。使用 save 方法锁定画布并完成操作之后,需要使用 restore 方法解除锁定
restore()	用于解锁定的画布,这种方法主要用在 save 方法之后,使用 save 方法锁定画布并完成操作后,需要使用 restore 方法解除锁定
clipRect (int left, int top, int right, int bottom)	用于裁剪画布,是设置画布的显示区域。在使用时,可以使用 Rect 对象来指定裁剪区,也可以通过指定矩形的 4 条边来指定裁剪区。该方法主要用于部分显示以及对画布中的部分对象进行操作的场合
rotate()	用于旋转画布,通过旋转画布,可以将画布上绘制的对象旋转。在使用这个方法时,将会把画布上的所有对象都旋转。为了只对某一个对象进行旋转,则可以通过 save 方法锁定画布,然后执行旋转操作,最后通过 restore 方法解锁,此后再绘制其他图形

在游戏开发中,可能需要对某个精灵执行旋转、缩放和一些其他操作。可以通过旋转画布来实现,但是旋转画布时会旋转画布上的所有对象,而只是需要旋转其中的一个,这时就需要用到 save 方法来锁定需要操作的对象,在操作之后通过 restore 方法来解除锁定。

下面通过一个实例来演示 Canvas 类的用法。

【例 7-2】 演示了怎样在 Android 中绘制基本的集合图形。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 Canvas_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件,文件采用默认代码。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
< fs.li6_3canvase.CanvasView
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"/>
</LinearLayout >
```

(3) 打开 res/Value 目录下的 strings.xml 文件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < string name = "app_name">画布实例</string >
    < string name = "hello_world"> Hello_world!</string >
```

```
        <string name = "action_settings">Settings</string>
        <string name = "circle">圆形</string>
        <string name = "square">正方形</string>
        <string name = "rect">长方形</string>
        <string name = "round_rect">圆角矩形</string>
        <string name = "oval">椭圆形</string>
        <string name = "triangle">三角形</string>
        <string name = "pentagon">五角形</string>
</resources>
```

(4) 打开 src\fs.com.cavnas_test 包下的 MainActivity.java 文件,在文件中实现跳转到主布局文件页面。代码为:

```
package fs.com.canvas_test;
import android.app.Activity;
import android.os.Bundle;
public class MainActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

(5) 在 src\fs.com.cavnas_test 包下的新建一个名为 CanvasView.java 的文件,在文件中的 Canvas 画布中绘制各种图形。代码为:

```
package fs.com.canvas_test;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Bitmap;
import android.graphics.Color;
import android.graphics.LinearGradient;
import android.graphics.Paint;
import android.graphics.Path;
import android.graphics.RectF;
import android.graphics.Shader;
import android.graphics.drawable.BitmapDrawable;
import android.util.AttributeSet;
import android.view.View;
public class CanvasView extends View
{
    public CanvasView(Context context,AttributeSet set)
    {
        super(context,set);
    }
    @Override
    //重写该方法,进行绘图
    protected void onDraw(Canvas canvas)
```

```
{
    super.onDraw(canvas);
    //把整张画布绘制成白色
    canvas.drawColor(Color.WHITE);
    Paint paint = new Paint();
    //去锯齿
    paint.setAntiAlias(true);
    paint.setColor(Color.BLUE);
    paint.setStyle(Paint.Style.STROKE);
    paint.setStrokeWidth(3);
    //绘制圆形
    canvas.drawCircle(40,40,30,paint);
    //绘制正方形
    canvas.drawRect(10,80,70,140,paint);
    //绘制矩形
    canvas.drawRect(10,150,70,190,paint);
    RectF re1 = new RectF(10,200,70,230);
    //绘制圆角矩形
    canvas.drawRoundRect(re1,15,15,paint);
    RectF re11 = new RectF(10,240,70,270);
    //绘制椭圆
    canvas.drawOval(re11,paint);
    //定义一个 Path 对象,封闭成一个三角形
    Path path1 = new Path();
    path1.moveTo(10,340);
    path1.lineTo(70,340);
    path1.lineTo(40,290);
    path1.close();
    //根据 Path 进行绘制,绘制三角形
    canvas.drawPath(path1,paint);
    //定义一个 Path 对象,封闭成一个五角形
    Path path2 = new Path();
    path2.moveTo(26,360);
    path2.lineTo(54,360);
    path2.lineTo(70,392);
    path2.lineTo(40,420);
    path2.lineTo(10,392);
    path2.close();
    //根据 Path 进行绘制,绘制五角形
    canvas.drawPath(path2,paint);
    //设置填充风格后绘制
    paint.setStyle(Paint.Style.FILL);
    paint.setColor(Color.RED);
    canvas.drawCircle(120,40,30,paint);
    //绘制正方形
    canvas.drawRect(90,80,150,140,paint);
    //绘制矩形
    canvas.drawRect(90,150,150,190,paint);
    RectF re2 = new RectF(90,200,150,230);
    //绘制圆角矩形
    canvas.drawRoundRect(re2,15,15,paint);
```

```
RectF re21 = new RectF(90,240,150,270);
//绘制椭圆
canvas.drawOval(re21,paint);
Path path3 = new Path();
path3.moveTo(90,340);
path3.lineTo(150,340);
path3.lineTo(120,290);
path3.close();
//绘制三角形
canvas.drawPath(path3,paint);
Path path4 = new Path();
path4.moveTo(106,360);
path4.lineTo(134,360);
path4.lineTo(150,392);
path4.lineTo(120,420);
path4.lineTo(90,392);
path4.close();
//绘制五角形
canvas.drawPath(path4,paint);
//设置渐变器后绘制
//为 Paint 设置渐变器
Shader mShader = new LinearGradient(0,0,40,60
    ,new int[] {
    Color.RED,Color.GREEN,Color.BLUE,Color.YELLOW }
    ,null,Shader.TileMode.REPEAT);
paint.setShader(mShader);
//设置阴影
paint.setShadowLayer(45,10,10,Color.GRAY);
//绘制圆形
canvas.drawCircle(200,40,30,paint);
//绘制正方形
canvas.drawRect(170,80,230,140,paint);
//绘制矩形
canvas.drawRect(170,150,230,190,paint);
RectF re3 = new RectF(170,200,230,230);
//绘制圆角矩形
canvas.drawRoundRect(re3,15,15,paint);
RectF re31 = new RectF(170,240,230,270);
//绘制椭圆
canvas.drawOval(re31,paint);
Path path5 = new Path();
path5.moveTo(170,340);
path5.lineTo(230,340);
path5.lineTo(200,290);
path5.close();
//根据 Path 进行绘制,绘制三角形
canvas.drawPath(path5,paint);
Path path6 = new Path();
path6.moveTo(186,360);
path6.lineTo(214,360);
path6.lineTo(230,392);
```

```
            path6.lineTo(200,420);
            path6.lineTo(170,392);
            path6.close();
            //根据 Path 进行绘制,绘制五角形
            canvas.drawPath(path6,paint);
            //设置字符大小后绘制
            paint.setTextSize(24);
            paint.setShader(null);
            Bitmap bitmap = null;
            //绘制 7 个字符串
            canvas.drawText(getResources().getString(R.string.circle),240,50,  paint);
            canvas.drawText(getResources().getString(R.string.square),240,120,paint);
            canvas.drawText(getResources().getString(R.string.rect),240,175,paint);
            canvas.drawText(getResources().getString(R.string.round_rect),230,220,paint);
            canvas.drawText(getResources().getString(R.string.oval),240,260,paint);
            canvas.drawText(getResources().getString(R.string.triangle),240,325,paint);
            canvas.drawText(getResources().getString(R.string.pentagon),240,390,paint);
           //显示图形
                bitmap = ((BitmapDrawable)getResources().getDrawable(R.drawable.ra)).getBitmap();
            canvas.drawBitmap(bitmap,50,450,null);  //绘制图像
        }
    }
```

运行程序,效果如图 7-2 所示。

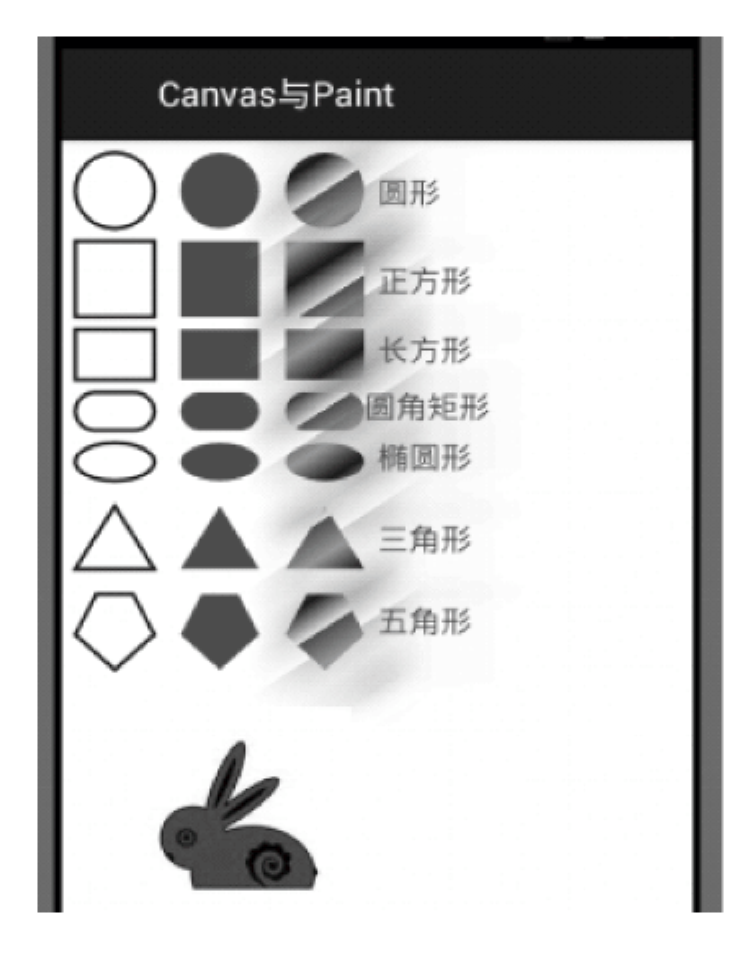


图 7-2　画布绘图

7.1.3　Bitmap 类

Bitmap 类代表位图,是 Android 系统中图像处理的一个重要类。使用该类,不仅可以获取图像文件信息进行图像剪切、旋转、缩放等操作,而且还可以指定格式保存图像文件。对于这些操作,都可以通过 Bitmap 类提供的方法来实现。Bitmap 类提供的常用方法如表 7-3 所示。

表 7-3　Bitmap 类的常用方法

方　　法	描　　述
compress(Bimtap.CompressFormat format, int quality,OutputStream stream)	用于将 Bitmap 对象压缩为指定格式并保存到指定的文件输出流中,其中 format 参数值可以是 Bitmap.Compress Format.PNG、Bitmap.CompressFormat.JPEG 和 Bitmap.CompressFormat.WEBP
createBitmap(Bitmap source,int x,int y,int width,int height,Matrix m,boolean filter)	用于从源位图的指定坐标点开始,"挖取"指定宽度和高度的一块图像来创建的 Bitmap 对象,并按 Matrix 指定规则进行变换
createBitmap(int width,int height,Bitmap.Config config)	用于创建一个指定宽度和高度的新的 Bitmap 对象
createBitmap(Bitmap source,int x,int y,int width,int height)	用于从源位图的指定坐标点开始,"挖取"指定宽度和高度的一块图像来创建新的 Bitmap 对象
createBitmap(int [] colors, int width, int height,Bitmap.Config config)	使用颜色数组创建一个指定宽度和高度的新的 Bitmap 对象,其中,数组元素的个数为 width * height
createBitmap(Bitmap src)	用于使用源位图创建一个新的 Bitmap 对象
createScaledBitmap(Bitmap src, int dstWidth, int dstHeight,boolean filter)	用于将源位图缩放为指定宽度和高度的新的 Bitmap 对象
isRecycled()	用于判断 Bitmap 对象是否被回收
recycle	强制回收 Bitmap 对象

例如,创建一个包括 4 个像素(每个像素对应一种颜色)的 Bitmap 对象的代码为:

```
Bitmap bitmap = Bitmap.createBitmap(new int{Color.RED, Color.GREEN, Color.BLUE, Color.
MAGENTA},4,2,Config.RGB_565);
```

7.1.4　BitmapFactory 类

在 Android 中,还提供了一个 BitmapFactory 类,该类为一个工具类,用于从不同的数据源来解析、创建 Bitmap 对象。BitmapFactory 类提供的创建 Bitmap 对象的常用方法如表 7-4 所示。

表 7-4　BitmapFactory 类的常用方法

方　　法	描　　述
decodeFile(String pathName)	用于从给定的路径所指定的文件中解析、创建 Bitmap 对象
decodeFileDescriptor(FileDescriptor fd)	用于从 FileDescriptor 对应的文件中解析、创建 Bitmap 对象
decodeResource(Resources res,int id)	用于根据给定的资源 ID 从指定的资源中解析、创建 Bitmap 对象
decodeStream(InputStream is)	用于从指定的输入流中解析、创建 Bitmap 对象

例如,要解析 SD 卡上的图片文件 img.jpg,并创建对应的 Bitmap 对象可以使用以下代码:

```
String path = "/sdcard/pictures/bccd/img.pg";
Bitmap bm = BitmapFactory.decodeFile(path);
```

要解析 Drawable 资源中保存的图片文件 img2.jpg，并创建对应的 Bitmap 对象可使用以下代码：

```
Bitmap bm = BitmapFactory.decodeResource(MainActivity.this.getResources(),R.drawable.img2);
```

下面通过一个实例来演示 Bitmap 类与 BitmapFactory 类的用法。

【例 7-3】 实现位图的显示处理。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Bitmap_test。

(2) res\layout 目录下的 main.xml 布局文件，采用默认代码。

(3) 打开 src\fs.bitmap_test 包下的 MainActivity.java 文件。代码为：

```
package fs.bitmap_test;
import android.app.Activity;
import android.os.Bundle;
public class MainActivity extends Activity
{
    private Surface_View bitmapView = null;
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        bitmapView = new Surface_View(this);
        setContentView(bitmapView);
    }
}
```

(4) 在 src\fs.bitmap_test 包下新建一个名为 Surface_View.java 的文件，用于实现位图操作。代码为：

```
package fs.bitmap_test;
import java.io.InputStream;
import android.content.Context;
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.drawable.BitmapDrawable;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
public class Surface_View extends SurfaceView
{
    //控制 surface 的接口，提供了控制 surface 的大小、格式、像素
    private SurfaceHolder surfaceHolder;
    private Canvas canvas = null;
    //x、y 用户触摸屏幕的坐标
    private float x = 0, y = 0;
    private Bitmap bitmap = null;
    public Surface_View(Context context)
    {
```

```
            super(context);
            //获取 SurfaceHolder 接口
            surfaceHolder = this.getHolder();
            //设置屏幕保持开启状态
            this.setKeepScreenOn(true);
            //获取资源文件 res
            Resources res = getResources();
            //获取位图资源文件的输入流
            InputStream inputStream = res.openRawResource(R.drawable.kc);
            //创建可绘制的位图对象
            BitmapDrawable bitmapDrawable = new BitmapDrawable(inputStream);
            //通过可绘制位图对象得到位图引用
            bitmap = bitmapDrawable.getBitmap();
            /*
              * 获取资源文件的引用 res:Resources res = getResources();
             * 获取图形资源文件
             * Bitmap bmp = BitmapFactory.decodeResource(res,R.drawable.h);
             */
        }
        //绘制位图
        private void onDraw()
        {
            try {
                //锁定 Canvas 画布
                canvas = surfaceHolder.lockCanvas();
                //设置 canvas 画布背景为黑色
                canvas.drawColor(Color.GREEN);
                //在画布上绘制位图
                //让位图的中心正好在触摸点位置上
            canvas.drawBitmap(bitmap,x - bitmap.getWidth()/2,y - bitmap.getHeight()/2,null);
            }
            catch (Exception ex) {
            }
            finally {
                if (canvas != null)
                    //解锁画布,并显示绘制图片
                    surfaceHolder.unlockCanvasAndPost(canvas);
            }
        }

        //触摸事件
         public boolean onTouchEvent(MotionEvent event)
        {
             x = event.getX();
             y = event.getY();
             onDraw();
            return true;
         }
    }
```

运行程序,效果如图 7-3 所示,可以利用鼠标拖动位图到画布上的任意位置中。



图 7-3 位图显示效果

7.2 Path 类

Android 提供的 Path 为一个非常有用的类，其可以预先在 View 上将 N 个点连成一条“路径”，然后调用 Canvas 的 drawPath（path，paint）即可沿着路径绘制图形。实际上 Android 还为路径绘制提供了 PathEffect 来定义绘制效果，PathEffect 包含了如下子类（每个子类代表一种绘制效果）：

- ComposePathEffect
- CornerPathEffect
- DashPathEffect
- DiscretePathEffect
- PathDashPathEffect
- SumPathEffect

下面通过一个实例来演示这几个路径的绘制效果。

【例 7-4】 绘制 7 条路径，分别演示了不使用效果和使用上面 6 种子类效果。其具体实现操作步骤为：

（1）在 Eclipse 中创建一个 Android 应用项目，命名为 Path_test。

（2）res\layout 目录下的 main.xml 布局文件，采用默认代码。

（3）打开 src\fs.path_test 包下的 MainActivity.java 文件，在文件中利用 Path 类实现提供的 7 个子类绘制的 7 种路径。代码为：

```
package fs.path_test;
import android.app.Activity;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.ComposePathEffect;
import android.graphics.CornerPathEffect;
import android.graphics.DashPathEffect;
import android.graphics.DiscretePathEffect;
import android.graphics.Paint;
import android.graphics.Path;
import android.graphics.PathDashPathEffect;
import android.graphics.SumPathEffect;
import android.graphics.PathEffect;
import android.os.Bundle;
import android.view.View;
public class MainActivity extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(new MyView(this));
    }
```

```
class MyView extends View
{
    float phase;
    PathEffect[] effects = new PathEffect[7];
    int[] colors;
    private Paint paint;
    Path path;
    public MyView(Context context)
    {
        super(context);
        paint = new Paint();
        paint.setStyle(Paint.Style.STROKE);
        paint.setStrokeWidth(4);
        //创建、并初始化 Path
        path = new Path();
        path.moveTo(0,0);
        for (int i = 1; i <= 15; i++)
        {
            //生成 15 个点,随机生成它们的 Y 坐标,并将它们连成一条 Path
            path.lineTo(i * 20,(float) Math.random() * 60);
        }
        //初始化 7 个颜色
        colors = new int[] {Color.BLACK,Color.BLUE,Color.CYAN
            ,Color.GREEN,Color.MAGENTA,Color.RED,  Color.YELLOW};
          }
    @Override
    protected void onDraw(Canvas canvas)
    {
        //将背景填充成白色
        canvas.drawColor(Color.WHITE);
        //------------下面开始初始化 7 种路径效果----------
        //不使用路径效果.
        effects[0] = null;
        //使用 CornerPathEffect 路径效果
        effects[1] = new CornerPathEffect(10);
        //初始化 DiscretePathEffect
        effects[2] = new DiscretePathEffect(3.0f,  5.0f);
        //初始化 DashPathEffect
        effects[3] = new DashPathEffect(new float[]
            { 20,10,5,10 },phase);
        //初始化 PathDashPathEffect
        Path p = new Path();
        p.addRect(0,0,8,8,Path.Direction.CCW);
        effects[4] = new PathDashPathEffect(p,12,phase,
            PathDashPathEffect.Style.ROTATE);
        //初始化 PathDashPathEffect
        effects[5] = new ComposePathEffect(effects[2],effects[4]);
        effects[6] = new SumPathEffect(effects[4],effects[3]);
        //将画布移动到(8,8)处开始绘制
        canvas.translate(8,8);
        //依次使用 7 种不同路径效果、7 种不同的颜色来绘制路径
```

```
                for (int i = 0; i < effects.length; i++)
                {
                    paint.setPathEffect(effects[i]);
                    paint.setColor(colors[i]);
                    canvas.drawPath(path, paint);
                    canvas.translate(0, 60);
                }
                //改变 phase 值,形成动画效果
                phase += 1;
                invalidate();
            }
        }
    }
```

运行程序,效果如图 7-4 所示。

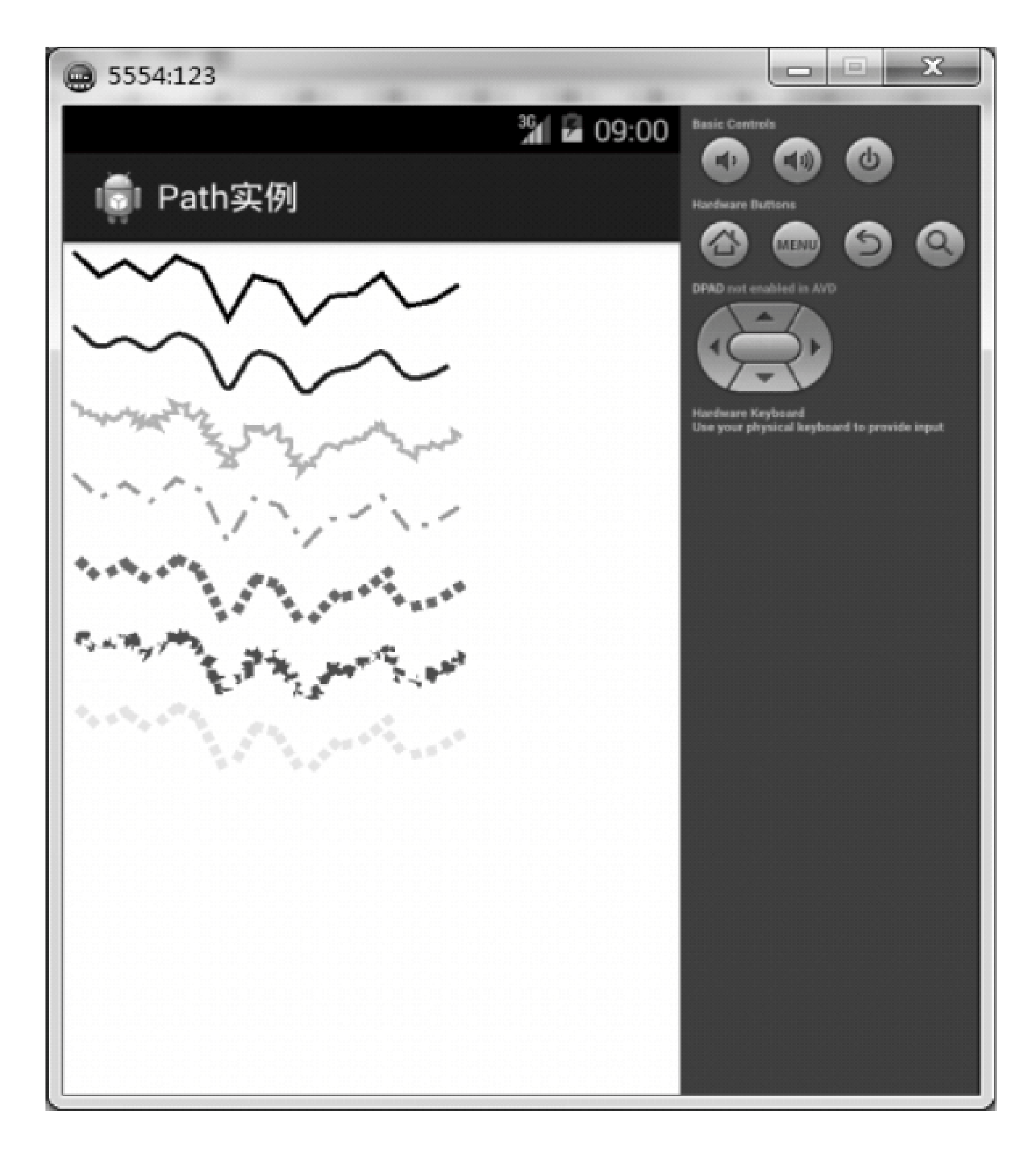


图 7-4　路径效果图

7.2.1　Android 绘制文本

通过以上程序中可看出,当定义 DashPathEffect、PathDashPath Effect 时可指定一个 phase 参数,该参数用于指定路径效果的相位,当该 phase 参数改变时,绘制效果也略有变化。上面的程序不停地改变 phase 参数,并不停地重绘该 View 组件,这将看到产生动画效果。

Path 类还包含一组矢量绘图方法,如画圆、矩形、弧、线条等。常用的绘图方法如表 7-5 所示。

表 7-5　Path 类的常用方法

方　　法	描　　述
addArc（RectF oval，float startAngle，float sweepAngle）	添加弧形路径
addCircle（float x，float y，float radius，Path. Direction dir）	添加圆形路径
addOval(RectF oval,Path. Direction dir)	添加椭圆形路径
addRect(RectF rect,Path. Direction dir)	添加矩形路径
addRoundRect（RectF rect，float rx，float ry，Path. Direction dir）	添加圆角矩形路径
moveTo(float x,float y)	设置开始绘制直线的起始点
lineTo(float x,float y)	在 moveTo()方法设置的起始点与该方法指定的结束点之间画一条直线，如果在调用该方法之前没有使用 moveTo()方法设置起始点，那么将从(0,0)点开始绘制直线
quadTo(float x1,float y1,float x2,float y2)	用于根据指定的参数绘制一条线段轨迹
close()	闭合路径

说明：在使用 addCircle()、addOval()、addRect()和 addRoundRect()方法时，需要指定 Path. Direction 类型的常量，可选值为 Path. Direction. CW（顺时针）和 Path. Direction. CCW（逆时针）。

例如，要创建一个顺时针旋转的圆形路径，可使用以下代码：

```
Path path = new Path();                          //创建并实例化一个 path 对象
path.addCircle(150,200,60,Path.Direction.CW);    //在 path 对象中添加一个圆形路径
```

要创建一个折线，可使用以下代码：

```
Path wpath = new Path();                         //创建并实例化一个 wpath 对象
wpath.moveTo(50,100);                            //设置起始点
wpath.lineTo(100,45);                            //设置第 1 段直线的结束点
wpath.lineTo(150,120);                           //设置第 2 段直线的线束点
wpath.lineTo(240,80);                            //设置第 3 段直线的结束点
```

要创建一个三角形路径，可使用以下代码：

```
Path path = new Path();
path.moveTo(50,50);
path.lineTo(100,10);
path.lineTo(150,50);
path.close();                                    //闭合路径
```

说明：在创建三角形路径时，如果不使用 close 方法闭合路径，那么绘制的将是两条线组成的折线。

下面通过一个实例来演示文本的绘制。

【例 7-5】 利用 drawTextOnPath 在 View 组件上绘制文本。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 drawText_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件,采用默认代码。代码为:

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#aabbcc">
    <TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "@string/hello_world" />
</RelativeLayout>
```

(3) 打开 src\fs.drawtext_test 包下的 MainActivity.java 文件,在文件中实现文本绘制。代码为:

```
package fs.drawtext_test;
import android.app.Activity;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Path;
import android.graphics.RectF;
import android.os.Bundle;
import android.view.View;
public class MainActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(new TextView(this));
    }
    class TextView extends View
    {
        final String DRAW_STR = "Android精要介绍";
        Path[] paths = new Path[3];
        Paint paint;
        public TextView(Context context)
        {
            super(context);
            paths[0] = new Path();
            paths[0].moveTo(0,0);
            for (int i = 1; i <= 7; i++)
            {
```

```
            //生成 7 个点,随机生成它们的 Y 坐标,并将它们连成一条 Path
            paths[0].lineTo(i * 30,(float) Math.random() * 30);
        }
        paths[1] = new Path();
        RectF rectF = new RectF(0,0,200,120);
        paths[1].addOval(rectF,Path.Direction.CCW);
        paths[2] = new Path();
        paths[2].addArc(rectF,60,180);
        //初始化画笔
        paint = new Paint();
        paint.setAntiAlias(true);
        paint.setColor(Color.RED);
        paint.setStrokeWidth(1);
    }
    @Override
    protected void onDraw(Canvas canvas)
    {
        canvas.drawColor(Color.WHITE);
        canvas.translate(40,40);
        //设置从右边开始绘制(右对齐)
        paint.setTextAlign(Paint.Align.RIGHT);
        paint.setTextSize(20);
        //绘制路径
        paint.setStyle(Paint.Style.STROKE);
        canvas.drawPath(paths[0],paint);
        //沿着路径绘制一段文本
        paint.setStyle(Paint.Style.FILL);
        canvas.drawTextOnPath(DRAW_STR,paths[0], - 8,20,paint);
        //画布下移 120
        canvas.translate(0,60);
        //绘制路径
        paint.setStyle(Paint.Style.STROKE);
        canvas.drawPath(paths[1],paint);
        //沿着路径绘制一段文本
        paint.setStyle(Paint.Style.FILL);
        canvas.drawTextOnPath(DRAW_STR,paths[1], - 20,20,paint);
        //画布下移 120
        canvas.translate(0,120);
        //绘制路径
        paint.setStyle(Paint.Style.STROKE);
        canvas.drawPath(paths[2],paint);
        //沿着路径绘制一段文本
        paint.setStyle(Paint.Style.FILL);
        canvas.drawTextOnPath(DRAW_STR,paths[2], - 10,20,paint);
    }
  }
}
```

运行程序,效果如图 7-5 所示。

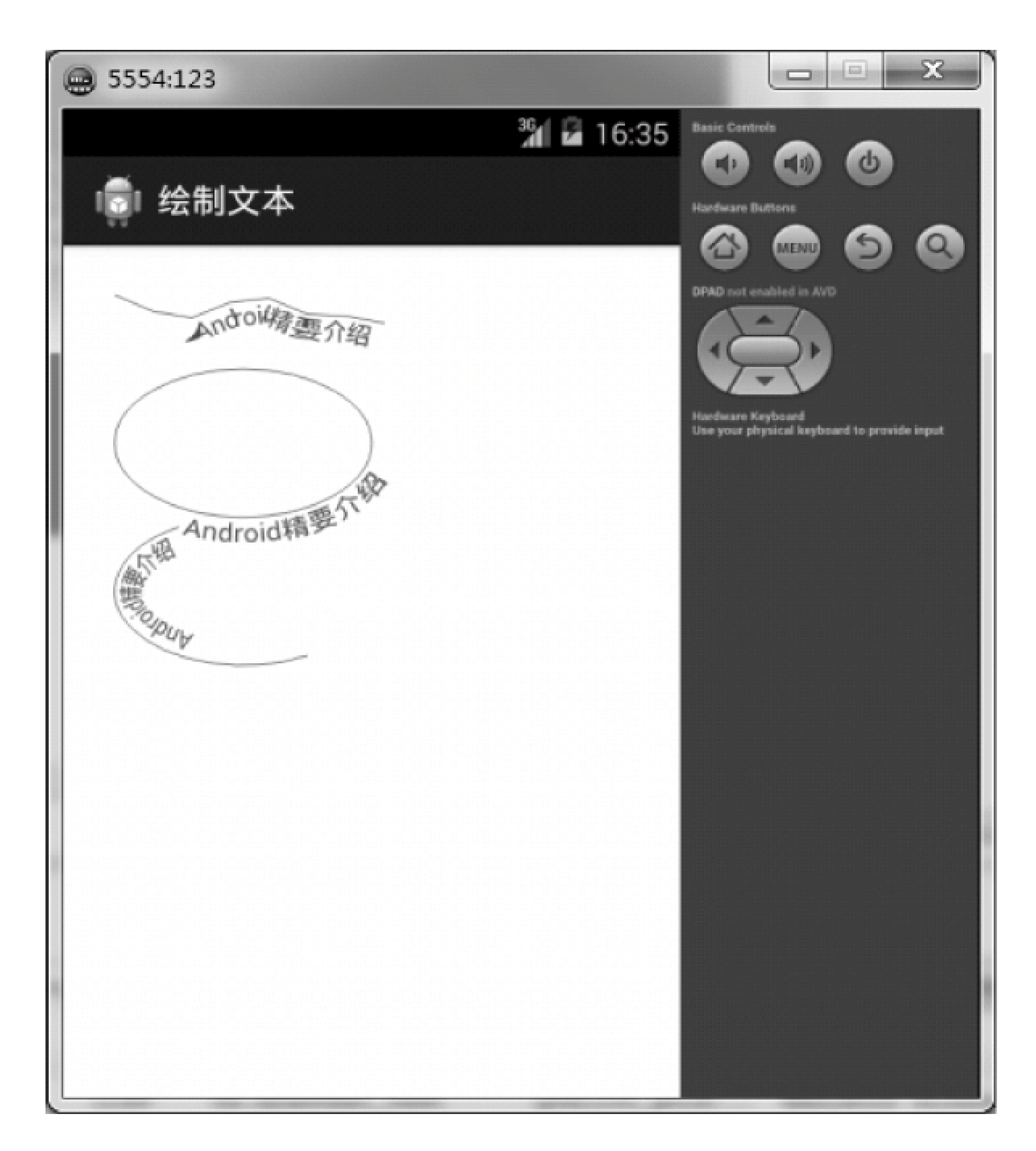


图 7-5　绘制文本

7.2.2　Android 绘制图片

在 Android 中，Canvas 类不仅可以绘制几何图形、文件和路径，还可以绘制图片。要想使用 Canvas 类绘制图片，只需要使用 Canvas 类提供的如表 7-6 所示的方法将 Bitmap 对象中保存的图片绘制到画面上即可。

表 7-6　Canvas 类提供的绘制图片的常用方法

方　　法	描　　述
drawBitmap(Bitmap bitmap,Rect src,RectF dst,Paint paint)	用于从指定点绘制从源位图中“挖取”的一块
drawBitmap(Bitmap bitmap,float left,float top,Paint paint)	用于在指定点绘制位图
drawBitmap(Bitmap bitmap,Rect src,Rect dst,Paint paint)	用于从指定点绘制从源位图中“挖取”的一块

例如，从源位图上“挖取”从(0,0)点到(450,280)点的一块图像，然后绘制到画布的(50,50)点到(400,300)点所指区域，可使用以下代码：

```
Rect src = new Rect(0,0,500,280);              //设置挖取的区域
Rect dst = new Rect(50,50,400,300);            //设置绘制的区域
canvas.drawBitmap(bm,src,dst,paint);           //绘制图片
```

下面通过一个实例来演示在 Android 中绘制图片。

【例 7-6】　在屏幕上绘制图片。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Canvasimage_test。

(2) 打开 res\layout 目录下 main.xml 布局文件，在文件中声明一个 ImageView 控件及两个 Button 控件。代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "@drawable/bj">
    <ImageView
        android:id = "@ + id/imageView1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignParentLeft = "true"
        android:layout_alignParentTop = "true"
        android:layout_marginLeft = "94dp"
        android:layout_marginTop = "114dp"
        android:src = "@drawable/ic_launcher" />
    <Button
        android:id = "@ + id/chooseButton"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignParentLeft = "true"
        android:layout_below = "@ + id/imageView1"
        android:layout_marginLeft = "24dp"
        android:layout_marginTop = "58dp"
        android:text = "选择按钮" />
    <Button
        android:id = "@ + id/saveButton"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignBaseline = "@ + id/chooseButton"
        android:layout_alignBottom = "@ + id/chooseButton"
        android:layout_marginLeft = "22dp"
        android:layout_toRightOf = "@ + id/imageView1"
        android:text = "保存按钮" />
</RelativeLayout>
```

(3) 打开 src\fs.canvasimage_test 包下的 MainActivity.java 文件，在文件中实现绘制图片。代码为：

```
package fs.canvasimage_test;
import java.io.FileNotFoundException;
import java.io.OutputStream;
import android.app.Activity;
import android.content.ContentValues;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.Bitmap.CompressFormat;
import android.graphics.BitmapFactory;
```

```java
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Matrix;
import android.graphics.Paint;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore.Images.Media;
import android.view.Display;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.View.OnTouchListener;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Toast;
public class MainActivity extends Activity implements OnTouchListener,OnClickListener {
    private ImageView image;
    private Paint paint;
    private Canvas canvas;
    private Bitmap bitmap;
    private Bitmap alterBitmap;
    private Button choose;
    private Button save;
    private final static int RESULT = 0;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        image = (ImageView) findViewById(R.id.imageView1);
        choose = (Button) findViewById(R.id.chooseButton);
        save = (Button)findViewById(R.id.saveButton);
        image.setOnTouchListener(this);
        choose.setOnClickListener(this);
        save.setOnClickListener(this);
    }
    private float downx = 0;
    private float downy = 0;
    private float upx = 0;
    private float upy = 0;
    public boolean onTouch(View v,MotionEvent event) {
        int action = event.getAction();
        switch (action) {
        case MotionEvent.ACTION_DOWN:
            downx = event.getX();
            downy = event.getY();
            break;
        case MotionEvent.ACTION_MOVE:
            //路径画板
            upx = event.getX();
            upy = event.getY();
            canvas.drawLine(downx,downy,upx,upy,paint);
```

```
                image.invalidate();
                downx = upx;
                downy = upy;
                break;
            case MotionEvent.ACTION_UP:
                //直线画板
                upx = event.getX();
                upy = event.getY();
                canvas.drawLine(downx,downy,upx,upy,paint);
                image.invalidate();                 //刷新
                break;
            default:
                break;
            }
            return true;
        }
        public void onClick(View arg0) {
            if(arg0 == choose){
            Intent intent = new Intent(Intent.ACTION_PICK,
                    android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
            startActivityForResult(intent,RESULT);
            }else if(arg0 == save){
                //保存画好的图片
                if(alterBitmap!= null){
                    try {
                        Uri imageUri = getContentResolver().insert(Media.EXTERNAL_CONTENT_URI,
new ContentValues());
                        OutputStream outputStream = getContentResolver().openOutputStream(imageUri);
                        alterBitmap.compress(CompressFormat.PNG,90,outputStream);
                        Toast.makeText(getApplicationContext(),"save!",Toast.LENGTH_SHORT).show();
                    } catch (FileNotFoundException e) {
                        e.printStackTrace();
                    }
                }
            }
        }
        @Override
        protected void onActivityResult(int requestCode,int resultCode,Intent data) {
            //TODO 自动存根法
            super.onActivityResult(requestCode,resultCode,data);
            if (resultCode == RESULT_OK) {
                Uri imageFileUri = data.getData();
                Display display = getWindowManager().getDefaultDisplay();
                float dw = display.getWidth();
                float dh = display.getHeight();
                try {
                    BitmapFactory.Options options = new BitmapFactory.Options();
                    options.inJustDecodeBounds = true;
                    bitmap = BitmapFactory.decodeStream(getContentResolver()
                            .openInputStream(imageFileUri),null,options);
                    int heightRatio = (int) Math.ceil(options.outHeight / dh);
```

```
                int widthRatio = (int) Math.ceil(options.outWidth / dw);
                if (heightRatio > 1 && widthRatio > 1) {
                    if (heightRatio > widthRatio) {
                        options.inSampleSize = heightRatio;
                    } else {
                        options.inSampleSize = widthRatio;
                    }
                }
                options.inJustDecodeBounds = false;
                bitmap = BitmapFactory.decodeStream(getContentResolver()
                        .openInputStream(imageFileUri),null,options);
                alterBitmap = Bitmap.createBitmap(bitmap.getWidth(),
                        bitmap.getHeight(),bitmap.getConfig());
                canvas = new Canvas(alterBitmap);
                paint = new Paint();
                paint.setColor(Color.GREEN);
                paint.setStrokeWidth(10);
                Matrix matrix = new Matrix();
                canvas.drawBitmap(bitmap,matrix,paint);
                image.setImageBitmap(alterBitmap);
                image.setOnTouchListener(this);
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            }
        }
    }
}
```

运行程序，效果如图 7-6 所示。



图 7-6　绘制图片

7.2.3 Path 类综合实例

本实例要实现一个画图板,并在画板上实现简易的涂鸦效果。当用户在触摸屏上移动时,即可在屏幕上绘制任意的图形。实现手绘功能其实是一种假象:表面上看起来可以随用户在触摸屏上自由地画曲线,实际上依然利用的是 Canvas 的 drawLine 方法画直线,每条直线都是从一次拖动事件发生点画到本次拖动事件发生点。当用户在触摸屏上移动时,两次拖动事件发生点的距离很小,多条极短的直线连接起来,肉眼看起来就是直线了。利用 Path 类,可非常方便地实现这种效果。

值得指出的是,如果程序每次都只是从上次拖动事件的发生点绘一条直线到本次拖动事件的发生点,那么用户前面绘制的就会丢失。为了保留用户之前绘制的内容,程序要借助于"双缓冲"技术。

"双缓冲"技术是指当程序需要在指定 View 上进行绘制时,程序并不直接绘制到该 View 组件上,而是先绘制到一个内存中的 Bitmap 图片(这就是缓冲)上,等到内存中的 Bitmap 绘制好后,再一次性地将 Bitmap 绘制到 View 组件上。

其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 DrawView_double。

(2) 打开 res\layout 目录下的 main.xml 文件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
     android:orientation = "vertical"
     android:layout_width = "fill_parent"
     android:layout_height = "fill_parent"
     android:background = "#aabbcc">
< fs.drawview_double.drawView
    android:id = "@ + id/drawView1"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent" />
</LinearLayout >
```

(3) 在 res\menu 目录下创建一个 menu_item.xml 的菜单源文件,该文件编写了实例中所应用的功能菜单。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< menu xmlns:android = "http://schemas.android.com/apk/res/android" >
    < item android:title = "@string/color">
        < menu >
            <! -- 定义一组单选菜单项 -- >
            < group android:checkableBehavior = "single" >
                <! -- 定义子菜单 -- >
                < item android:id = "@ + id/red" android:title = "@string/color_red"/>
                < item android:id = "@ + id/green" android:title = "@string/color_green"/>
                < item android:id = "@ + id/blue" android:title = "@string/color_blue"/>
            </group >
        </menu >
    </item >
    < item android:title = "@string/width">
        < menu >
```

```
            <! -- 定义子菜单 -->
            <group>
                <item android:id="@+id/width_1" android:title="@string/width_1"/>
                <item android:id="@+id/width_2" android:title="@string/width_2"/>
                <item android:id="@+id/width_3" android:title="@string/width_3"/>
            </group>
        </menu>
    </item>
    <item android:id="@+id/clear" android:title="@string/clear"/>
     <item android:id="@+id/save" android:title="@string/save"/>
</menu>
```

(4) 打开 res\layout 目录下的 strings.xml 文件，在该文件中实现变量的赋值。代码为：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">涂鸦</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
     <string name="width_1">1 像素</string>
     <string name="width_2">5 像素</string>
     <string name="width_3">10 像素</string>
     <string name="color_red">红色</string>
     <string name="color_green">绿色</string>
     <string name="color_blue">蓝色</string>
     <string name="color">画笔颜色</string>
     <string name="width">画笔宽度</string>
     <string name="clear">擦除绘画</string>
     <string name="save">保存绘画</string>
</resources>
```

(5) 打开 src\fs.drawview_double 包下的 MainActivity.java 文件，该程序实现加载、显示布局，加载、显示菜单资源，并为各菜单项编写事件响应。代码为：

```
package fs.drawview_double;
import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    //创建选项菜单
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflator = new MenuInflater(this);    //实例化一个 MenuInflater 对象
        inflator.inflate(R.menu.menu_item,menu);           //解析菜单文件
        return super.onCreateOptionsMenu(menu);
    }
```

```
    //当菜单项被选择时,做出相应的处理
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
drawView dv = (drawView) findViewById(R.id.drawView1);     //获取自定义的绘图视图
        dv.paint.setXfermode(null);                        //取消擦除效果
        dv.paint.setStrokeWidth(1);                        //初始化画笔的宽度
        switch (item.getItemId()) {
        case R.id.red:
            dv.paint.setColor(Color.RED);                  //设置画笔的颜色为红色
            item.setChecked(true);
            break;
        case R.id.green:
            dv.paint.setColor(Color.GREEN);                //设置画笔的颜色为绿色
            item.setChecked(true);
            break;
        case R.id.blue:
            dv.paint.setColor(Color.BLUE);                 //设置画笔的颜色为蓝色
            item.setChecked(true);
            break;
        case R.id.width_1:
            dv.paint.setStrokeWidth(1);                    //设置笔触的宽度为 1 像素
            break;
        case R.id.width_2:
            dv.paint.setStrokeWidth(5);                    //设置笔触的宽度为 5 像素
            break;
        case R.id.width_3:
            dv.paint.setStrokeWidth(10);                   //设置笔触的宽度为 10 像素
            break;
        case R.id.clear:
            dv.clear();                                    //擦除绘画
            break;
        case R.id.save:
            dv.save();                                     //保存绘画
            break;
        }
        return true;
    }
}
```

(6) 在 src\drawview_double 包下创建一个 drawView.java 文件,在文件中自定义 View 组件,重写该 View 的 onDraw(Canvas canvas)方法,并实现画板的涂鸦功能。代码为:

```
package fs.drawview_double;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.Bitmap.Config;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Path;
import android.graphics.PorterDuff;
```

```java
import android.graphics.PorterDuffXfermode;
import android.util.AttributeSet;
import android.view.MotionEvent;
import android.view.View;
public class drawView extends View {
    private int view_width = 0;                //屏幕的宽度
    private int view_height = 0;               //屏幕的高度
    private float preX;                        //起始点的 x 坐标值
    private float preY;                        //起始点的 y 坐标值
    private Path path;                         //路径
    public Paint paint = null;                 //画笔
    Bitmap cacheBitmap = null;                 //定义一个内存中的图片,该图片将作为缓冲区
    Canvas cacheCanvas = null;                 //定义 cacheBitmap 上的 Canvas 对象
    //构造方法
    public drawView(Context context,AttributeSet set) {
        super(context,set);
        //获取屏幕的宽度
        view_width = context.getResources().getDisplayMetrics().widthPixels;
        //获取屏幕的高度
        view_height = context.getResources().getDisplayMetrics().heightPixels;
        System.out.println(view_width + "*" + view_height);
        //创建一个与该 View 相同大小的缓存区
        cacheBitmap = Bitmap.createBitmap(view_width,view_height,Config.ARGB_8888);
        cacheCanvas = new Canvas();
        path = new Path();
        cacheCanvas.setBitmap(cacheBitmap);  //在 cacheCanvas 上绘制 cacheBitmap
        paint = new Paint(Paint.DITHER_FLAG);
        paint.setColor(Color.RED);             //设置默认的画笔颜色
        //设置画笔风格
        paint.setStyle(Paint.Style.STROKE);      //设置填充方式为描边
        paint.setStrokeJoin(Paint.Join.ROUND);   //设置笔刷的图形样式
        paint.setStrokeCap(Paint.Cap.ROUND);     //设置画笔转弯处的连接风格
        paint.setStrokeWidth(1);                 //设置默认笔触的宽度为 1 像素
        paint.setAntiAlias(true);                //使用抗锯齿功能
        paint.setDither(true);                   //使用抖动效果
    }
    //重写 onDraw()方法
    @Override
    public void onDraw(Canvas canvas) {
        canvas.drawColor(0xFFFFFFFF);            //设置背景颜色
        Paint bmpPaint = new Paint();            //采用默认设置创建一个画笔
        canvas.drawBitmap(cacheBitmap,0,0,bmpPaint); //绘制 cacheBitmap
        canvas.drawPath(path,paint);                 //绘制路径
        canvas.save(Canvas.ALL_SAVE_FLAG);           //保存 canvas 的状态
        canvas.restore();
        //恢复 canvas 之前保存的状态,防止保存后对 canvas 执行的操作对后续的绘制有影响
    }
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        //获取触摸事件的发生位置
        float x = event.getX();
        float y = event.getY();
        switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            path.moveTo(x,y);             //将绘图的起始点移到(x,y)坐标点的位置
            preX = x;
            preY = y;
```

```
                break;
            case MotionEvent.ACTION_MOVE:
                float dx = Math.abs(x - preX);
                float dy = Math.abs(y - preY);
                if (dx >= 5 || dy >= 5) {    //判断是否在允许的范围内
                    path.quadTo(preX,preY,(x + preX) / 2,(y + preY) / 2);
                    preX = x;
                    preY = y;
                }
                break;
            case MotionEvent.ACTION_UP:
                cacheCanvas.drawPath(path,paint); //绘制路径
                path.reset();
                break;
            }
            invalidate();
            return true;                     //返回 true 表明处理方法已经处理了该事件
        }
        public void clear() {
            paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.CLEAR));
            paint.setStrokeWidth(50);          //设置笔触的宽度
        }
        public void save() {
            try {
                saveBitmap("myPicture");
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        //保存绘制好的位图
        public void saveBitmap(String fileName) throws IOException {
            File file = new File("/sdcard/pictures/" + fileName + ".png");//创建文件对象
            file.createNewFile();                                  //创建一个新文件
        FileOutputStream fileOS = new FileOutputStream(file);  //创建一个文件输出流对象
            cacheBitmap.compress(Bitmap.CompressFormat.PNG,100,fileOS);
            //将绘图内容压缩为 PNG 格式输出到输出流对象中
            fileOS.flush();         //将缓冲区中的数据全部写出到输出流中
            fileOS.close();         //关闭文件输出流对象
        }
    }
```

(7) 打开 AndroidManifest.xml 文件，用于为程序设计权限。代码为：

```
<?xml version = "1.0" encoding = "utf-8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.drawview_double"
    android:versionCode = "1"
    android:versionName = "1.0" >
    <uses-sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
    <!-- 设置权限 -->
        <uses-permission android:name = "android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
    <uses-permission android:name = "android.permission.WRITE_EXTERNAL_STORAGE"/>
    <application
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
```

```
            android:label = "@string/app_name"
            android:theme = "@style/AppTheme" >
            <activity
                android:name = ".MainActivity"
                android:label = "@string/app_name" >
                <intent - filter>
                    <action android:name = "android.intent.action.MAIN" />
                    <category android:name = "android.intent.category.LAUNCHER" />
                </intent - filter>
            </activity>
        </application>
    </manifest>
```

运行程序,默认效果如图 7-7(a)所示,当单击界面中右上角的按钮时,显示选项菜单,效果如图 7-7(b)所示,选择菜单项后弹出对应的子菜单,效果如图 7-7(c)所示,当完成菜单项设置后,可实现涂鸦效果,如图 7-7(d)所示。



(a) 默认界面

(b) 选项菜单



(c) 选项子菜单

(d) 涂鸦效果

图 7-7　画图板

7.3 Android 3D 图形

在现在这个网络游戏逐渐盛行的时代，2D 游戏已经不能完全满足用户的需求，3D 技术已经被广泛地应用在 PC 游戏中，3D 技术下一步将向手机平台发展，而 Android 作为当前最流行的手机操作系统，完全内置 3D 技术——OpenGL 支持。

7.3.1 OpenGL 概述

OpenGL(Open Graphics Library)是由 SGI 公司于 1992 年发布的，一个功能强大、调用方便的底层图形库，它为编程人员提供了统一的操作，以便充分利用任何制造商提供的硬件。OpenGL 的核心实现了视区和光照等人们所熟知的概念，并试图向开发人员隐藏大部分硬件层。

由于 OpenGL 是专门为工作站设计的，它太大了，无法安装在移动设备上。所以 Khronos Group 为 OpenGL 提供了一个子集 OpenGL ES(OpenGL for Embedded System)。OpenGL ES 是免费的、跨平台的、功能完善的 2D/3D 图形库接口 API，它专门针对多种嵌入式系统(包括手机、PDA 和游戏主机等)而设计的，提供一种标准方法来描述在图形处理器或主 CPU 上渲染这些图像的底层硬件。

在 PC 领域，一直有两种标准的 3D API 在进行竞争，OpenGL 和 DirectX。一般主流的游戏和显卡都支持这两种渲染方式，DirectX 在 Windows 平台上有很大的优势，但是 OpenGL 具有更好的跨平台性。

由于嵌入式系统和 PC 相比，整体上讲，CPU、内存等都比 PC 差得很多，而且对能耗有着特殊的要求，许多嵌入式设备并没有浮点运算协处理器，针对嵌入式系统的以上特点，Khronos 对标准的 OpenGL 系统进行了维护和改动，以期望满足嵌入式设备对 3D 绘图的要求。

7.3.2 Android 构建 3D 图形

在 Android 中使用 GLSurfaceView 来显示 OpenGL 视图，该类位于 android.opengl 包里面。它提供了一个专门用于渲染 3D 的接口 Renderer。接下来就来一步步构建自己的 Renderer 类。

(1) 为 Renderer 类载入命名空间。代码为：

```
import android.opengl.GLSurfaceView.Renderer;
```

(2) 新建一个类来实现 Renderer 接口。代码为：

```
public class ThreeDGl implements Renderer
{ }
```

(3) 如上代码所写，程序实现了 Renderer 类，则必须重写以下方法：

```
public void onDrawFrame(GL10 gl)
{
}
```

```
public void onSurfaceChanged(GL10 gl,int width,int height)
{}
public void onSurfaceCreated(GL10 gl,EGLConfig config)
{}
```

(4) 当窗口被创建时需要调用 onSurfaceCreate,可以在这里对 OpenGL 做一些初始化工作,例如:

```
  //启用阴影平滑
gl.glShadeModel(GL10.GL_SMOOTH);
   //黑色背景
   gl.glClearColor(0,0,0,0);
   //设置深度缓存
   gl.glClearDepthf(1.0f);
   //启用深度测试
   gl.glEnable(GL10.GL_DEPTH_TEST);
   //所作深度测试的类型
   gl.glDepthFunc(GL10.GL_LEQUAL);
   //告诉系统对透视进行修正
   gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,GL10.GL_FASTEST);
```

GL10 提供的用于进行初始化的方法如下所示:

- glHint:用于告诉 OpenGL 希望进行最好的透视修正,这会轻微地影响性能,但会使得透视图更好看。
- glClearColor:设置清除屏幕时所用的颜色,色彩值的范围为 0.0f~1.0f,即颜色从暗到亮的过程。
- glShadeModel:用于启用阴影平滑度。阴影平滑通过多边形精细地混合色彩,并对外部光进行平滑。
- glDepthFunc:将深度缓存设想为屏幕后面的层,它不断地对物体进入屏幕内部的深度进行跟踪。
- glEnable:启用深度测试。

(5) 当窗口大小发生改变时系统将调用 onSurfaceChange 方法,可以在该方法中设置 OpenGL 场景大小。代码为:

```
//设置 OpenGL 场景的大小
gl.glViewport(0,0,width,height);
```

(6) 场景画出来以后就要实现场景里面的内容,例如:设置它的透视图,让它有种越远的东西看起来越小的感觉。代码为:

```
//设置投影矩阵
        gl.glMatrixMode(GL10.GL_PROJECTION);
        //重置投影矩阵
        gl.glLoadIdentity();
        //设置视口的大小
        gl.glFrustumf(-ratio,ratio,-1,1,1,10);
        //选择模型观察矩阵
        gl.glMatrixMode(GL10.GL_MODELVIEW);
```

```
    //重置模型观察矩阵
    gl.glLoadIdentity();
```

以上代码中,主要语句的用法为:

- gl. glMatrixMode(GL10. GL_PROJECTION):指明接下来的代码将影响projection matrix(投影矩阵),投影矩阵负责为场景增加透视度。
- gl. glLoadIdentity():此方法相当于手机的重置功能,它将所选择的矩阵状态恢复成原始状态,调用 glLoadIdentity()之后为场景设置透视图。
- gl. glMatrixMode(GL10. GL_MODELVIEW):指明任何新的变换都将会影响modelview matrix(模型观察矩阵)。
- gl. glFrustumf(-ratio,ratio,-1,1,1,10):此方法,前面 4 个参数用于确定窗口的大小,而后面两个参数分别是在场景中所能绘制深度的起点和终点。

(7) 了解了上面两个重写方法的作用和功能之后,第 3 个方法 onDrawFrame 从字面上理解就知道此方法是做绘制图操作的。在绘图之前,需要将屏幕清除成前面所指定的颜色,清除尝试缓存并且重置场景,然后就可以绘图了。代码为:

```
//清除屏幕和深度缓存
gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
//重置当前的模型观察矩阵
gl.glLoadIdentity();
```

(8) Renderer 类在实现了上面的 3 个重写之后,在程序入口中只需要调用。代码为:

```
Renderer render = new ThreeDGl(this);
    /** 第一次调用 activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        GLSurfaceView gview = new GLSurfaceView(this);
        gview.setRenderer(render);
        setContentView(gview);
    }
```

下面通过一个实例来演示 GL10 的用法。

【例 7-7】 利用 glDrawElements 方法绘制 6 个面采用不同颜色的立方体。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 GL10_3D。

(2) 打开 res\layout 目录下的 main.xml 布局文件,文件代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
<TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
```

```
    android:text = "@string/hello_world"/>
</LinearLayout>
```

(3) 打开 src\fs.gl10_3d 包下的 MainActivity.java 文件,在文件中创建一个 GLSurfaceView 对象,并为该对象指定 Renderer 对象,再设置 Activity 显示内容为 GLSurfaceView 对象。代码为:

```
package fs.gl10_3d;
import android.app.Activity;
import android.opengl.GLSurfaceView;
import android.os.Bundle;
public class MainActivity extends Activity {
    private GLSurfaceView mGLView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mGLView = new GLSurfaceView(this);//创建一个 GLSurfaceView 对象
         //为 GLSurfaceView 指定使用的 Renderer 对象
        mGLView.setRenderer(new Renderertest());
            setContentView(mGLView);      //设置 Activity 显示的内容为 GLSurfaceView 对象
    }
    @Override
    protected void onResume() {
        super.onResume();
        mGLView.onResume();
    }
    @Override
    protected void onPause() {
        super.onPause();
        mGLView.onPause();
    }
}
```

(4) 在 src\fs.gl10_3d 包下新建一个 Rendertest.java 文件,在文件中实现自定义 Render 类。代码为:

```
package fs.gl10_3d;
import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;
import android.opengl.GLSurfaceView;
import android.opengl.GLU;
public class Renderertest implements GLSurfaceView.Renderer {
    private final GLCube cube;                       //立方体对象
    public Renderertest() {
        cube = new GLCube();                         //实例化立方体对象
    }
    public void onSurfaceCreated(GL10 gl,EGLConfig config) {
        gl.glClearColor(0.7f,0.9f,0.9f,1.0f);          //设置窗体背景颜色
        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);  //启用顶点坐标数组
        gl.glDisable(GL10.GL_DITHER);                  //关闭抗抖动
        gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,GL10.GL_FASTEST);
```

```
            //设置系统对透视进行修正
            gl.glShadeModel(GL10.GL_SMOOTH);                //设置阴影平滑模式
            gl.glEnable(GL10.GL_DEPTH_TEST);                //启用深度测试
            gl.glDepthFunc(GL10.GL_LEQUAL);                 //设置深度测试的类型
        }
        public void onDrawFrame(GL10 gl) {
            gl.glClear(GL10.GL_COLOR_BUFFER_BIT |GL10.GL_DEPTH_BUFFER_BIT);
                                                            //清除颜色缓存和深度缓存
            gl.glMatrixMode(GL10.GL_MODELVIEW);             //设置使用模型矩阵进行变换
            gl.glLoadIdentity();                            //初始化单位矩阵
            //当使用 GL_MODELVIEW 模式时,必须设置视点,也就是观察点
            GLU.gluLookAt(gl,0,0, - 5,0f,0f,0f,0f,1.0f,0.0f);
            gl.glRotatef(1000, - 0.1f, - 0.1f,0.05f);       //旋转总坐标系
            //绘制立方体
            cube.draw(gl);
        }
        public void onSurfaceChanged(GL10 gl,int width,int height) {
            gl.glViewport(0,0,width,height);                //设置 OpenGL 场景的大小
            float ratio = (float) width / height;           //计算透视视窗的宽度、高度比
            gl.glMatrixMode(GL10.GL_PROJECTION);            //将当前矩阵模式设为投影矩阵
            gl.glLoadIdentity();                            //初始化单位矩阵
            //设置透视视窗的空间大小
            GLU.gluPerspective(gl,45.0f,ratio,1,100f);
        }
    }
```

(5) 在 src\fs.gl10_3d 包下新建一个 GLCube.java 文件,在文件中实现绘制不同颜色的立方体。代码为:

```
package fs.gl10_3d;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.IntBuffer;
import javax.microedition.khronos.opengles.GL10;
/**
 * 定义一个简单的模型:立方体
 */
public class GLCube {
    private final IntBuffer mVertexBuffer;              //顶点坐标数据缓冲
    public GLCube() {
        int one = 65536;
        int half = one / 2;
        int vertices[] = {
                //前面
                - half, - half,half,half, - half,half,
                - half,half,half,half,half,half,
                //背面
                - half, - half, - half, - half,half, - half,
                half, - half, - half,half,half, - half,
                //左面
                - half, - half,half, - half,half,half,
```

```
                -half,-half,-half,-half,half,-half,
                //右面
                half,-half,-half,half,half,-half,
                half,-half,half,half,half,half,
                //上面
                -half,half,half,half,half,half,
                -half,half,-half,half,half,-half,
                //下面
                -half,-half,half,-half,-half,-half,
                half,-half,half,half,-half,-half,
            };                                  //定义顶点位置
        //创建顶点坐标数据缓冲
        ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length * 4);
        vbb.order(ByteOrder.nativeOrder());     //设置字节顺序
        mVertexBuffer = vbb.asIntBuffer();      //转换为 int 型缓冲
        mVertexBuffer.put(vertices);            //向缓冲中放入顶点坐标数据
        mVertexBuffer.position(0);              //设置缓冲区的起始位置
    }
    public void draw(GL10 gl) {
gl.glVertexPointer(3,GL10.GL_FIXED,0,mVertexBuffer);   //为画笔指定顶点坐标数据
        //绘制 FRONT 和 BACK 两个面
        gl.glColor4f(1,0,0,1);
        gl.glNormal3f(0,0,1);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP,0,4);    //绘制图形
        gl.glColor4f(1,0,0.5f,1);
        gl.glNormal3f(0,0,-1);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP,4,4);    //绘制图形
        //绘制 LEFT 和 RIGHT 两个面
        gl.glColor4f(0,1,0,1);
        gl.glNormal3f(-1,0,0);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP,8,4);    //绘制图形
        gl.glColor4f(0,1,0.5f,1);
        gl.glNormal3f(1,0,0);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP,12,4);   //绘制图形
        //绘制 TOP 和 BOTTOM 两个面
        gl.glColor4f(0,0,1,1);
        gl.glNormal3f(0,1,0);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP,16,4);   //绘制图形
        gl.glColor4f(0,0,0.5f,1);
        gl.glNormal3f(0,-1,0);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP,20,4);   //绘制图形
    }
}
```

(6) 打开 AndroidManifest.xml 文件,用于设置权限。代码为:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="fs.gl10_3d"
    android:versionCode="1"
    android:versionName="1.0" >
```

```
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />
    <!-- 设置权限 -->
<uses-feature android:glEsVersion="0x00020000" android:required="true" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="fs.gl10_3d.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

运行程序，得到的立方体如图 7-8 所示。

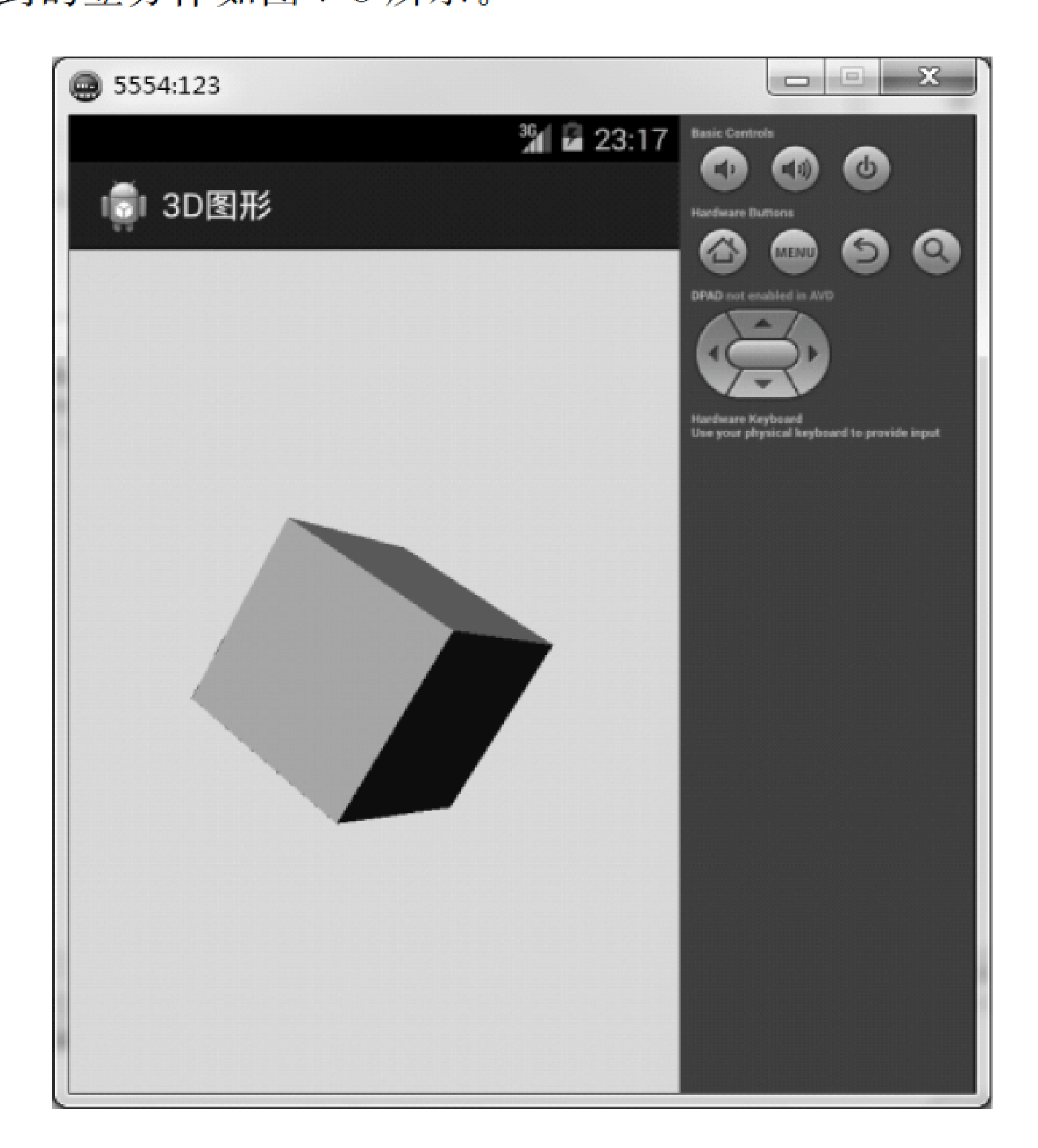


图 7-8　立方体

7.3.3　Android 纹理贴图

为了让 3D 图形更加逼真，需要为这些 3D 图形应用纹理贴图。为了在 OpenGL ES 中启用纹理贴图功能，可以在 Redderer 实现类的 onSurfaceCreated(GL10 gl, EGLConfig

config)方法中启动纹理贴图。代码为：

```
//启动 2D 纹理贴图
gl.glEnable(GL10.GL_TEXTURE_2D);
```

接着就需要准备一张图片来作为纹理贴图了，建议该图片的长宽为 2 的 *N* 次方，例如长、宽分为 256、512 等。把这张准备贴图的位图放在 Android 项目的 res/drawable_mdpi 目录下，方便应用程序加载该图片资源。

接着记载该图片并生成对应的纹理贴图。代码为：

```
//加载位图
bitmap = BitmapFactory.decodeResource(context.getResources(),R.drawable.send);
int [] textures = new int[1];
//指定生成 N 个纹理(第 1 个参数指定生成 1 个纹理)
//textures 数组将负责存储所有纹理的代号
gl.glGenTextures(1,textures,0);
//获取 textures 纹理数组中的第 1 个纹理
texture = textures[0];
//通知 OpenGL 将 texture 纹理绑定到 GL10.GL_TEXTURE_2D 目标中
gl.glBindTexture(GL10.GL_TEXTURE_2D,texture);
//设置纹理被缩小(距离视点很远时被缩小)时候的滤波方式
gl.glTexParameterf(GL10.GL_TEXTURE_2D,GL10.GL_TEXTURE_MIN_FILTER,GL10.GL_NEAREST);
//设置纹理被放大(距离视点很近时被放大)时候的滤波方式
gl.glTexParameterf(GL10.GL_TEXTURE_2D,GL10.GL_TEXTURE_MAG_FILTER,GL10.GL_LINEAR);
//设置在横向、纵向上都是平铺纹理
gl.glTexParameterf(GL10.GL_TEXTURE_2D,GL10.GL_TEXTURE_WRAP_S,GL10.GL_REPEAT);
gl.glTexParameterf(GL10.GL_TEXTURE_2D,GL10.GL_TEXTURE_WRAP_T,GL10.GL_REPEAT);
//加载位图生成纹理
GLUtils.textIamge2D(GL10.GL_TEXTURE_2D,0,bitmap,0);
```

在以上程序中设置了当纹理被放大时使用 GL10.GL_LINEAR 滤波方式；当纹理被缩小时使用 GL10.GL_NEAREST 滤波方式；一般来说，使用 GL10.GL_LINEAR 滤波方式有较好的效果，但系统开销略微大了一些，具体采用哪种滤波方式则取决于目录机器本身的性能。

以上程序和 GL10 使用如下方法。

- glGenTextures(int n,int[] textures,int offset)：该方法指定一次性生成 *n* 个纹理，该方法所生成的纹理的代号放入 textures 数组中，offset 指定从第几个数组元素开始存放编号代码。
- glBindTexture(int target,int texture)：该方法用于将 texture 纹理绑定到 target 目录上。
- glTexParameterf(int target,int pname,float param)：该方法用于为 target 纹理目标设置属性，其中，第 2 个参数为属性名，第 3 个参数为属性值。

在 3D 绘制中进行纹理贴图也很简单，只需要 3 步：

- 设置启用贴图坐标数组。
- 设置贴图坐标的数组信息。
- 调用 GL10 的 glBindTexture(int target,int texture)方法执行贴图。

下面通过一个实例来演示 3D 纹理贴图。

【例 7-8】 为例 7-7 所绘制的立方体进行纹理贴图。其具体实现步骤为：

(1) res\layout 目录下的 main. xml 布局文件及 res\value 目录下的 strings. xml 文件，采用与例 7-7 相同的代码。

(2) 打开 src\fs. gl10_3d 包下的 MainActivity. java 文件，在文件中创建一个 GLSurfaceView 对象，并为该对象指定 Renderer 对象，再设置 Activity 显示内容为 GLSurfaceView 对象。代码为：

```
package com.mingrisoft;
package fs.gl10_3d;
import android.app.Activity;
import android.opengl.GLSurfaceView;
import android.os.Bundle;
public class MainActivity extends Activity {
    private GLSurfaceView mGLView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mGLView = new GLSurfaceView(this);              //创建一个 GLSurfaceView 对象
        //为 GLSurfaceView 指定使用的 Renderer 对象
        mGLView.setRenderer(new Renderertest(this));
            setContentView(mGLView);
            //设置 Activity 显示的内容为 GLSurfaceView 对象
    }
    @Override
    protected void onResume() {
        super.onResume();
        mGLView.onResume();
    }
    @Override
    protected void onPause() {
        super.onPause();
        mGLView.onPause();
    }
}
```

(3) 打开 src\fs. gl10_3d 包下的 Renderertest. java 文件，在例 7-7 的基础上实例化对象。代码为：

```
package fs.gl10_3d;
import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;
import android.content.Context;
import android.opengl.GLSurfaceView;
import android.opengl.GLU;
public class Renderertest implements GLSurfaceView.Renderer {
    private final GLCube cube;                          //立方体对象
    private Context context;
    public Renderertest(Context context) {
```

```
        cube = new GLCube();                        //实例化立方体对象
        this.context = context;
    }
    public void onSurfaceCreated(GL10 gl,EGLConfig config) {
        gl.glClearColor(0.7f,0.9f,0.9f,1.0f);       //设置窗体背景颜色
        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);  //启用顶点坐标数组
        gl.glDisable(GL10.GL_DITHER);               //关闭抗抖动
        gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,GL10.GL_FASTEST);
                                                    //设置系统对透视进行修正
        gl.glShadeModel(GL10.GL_SMOOTH);            //设置阴影平滑模式
        gl.glEnable(GL10.GL_DEPTH_TEST);            //启用深度测试
        gl.glDepthFunc(GL10.GL_LEQUAL);             //设置深度测试的类型
        /** ============ 应用纹理贴图 =============== */
gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);  //启用贴图坐标数组
        gl.glEnable(GL10.GL_TEXTURE_2D);            //启用纹理贴图
        cube.loadTexture(gl,context,R.drawable.bj2);  //进行纹理贴图
    }
    public void onDrawFrame(GL10 gl) {
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
        //清除颜色缓存和深度缓存
        gl.glMatrixMode(GL10.GL_MODELVIEW);         //设置使用模型矩阵进行变换
        gl.glLoadIdentity();                        //初始化单位矩阵
        //当使用 GL_MODELVIEW 模式时,必须设置视点,也就是观察点
        GLU.gluLookAt(gl,0,0,-5,0f,0f,0f,0f,1.0f,0.0f);
        gl.glRotatef(1000,-0.1f,-0.1f,0.05f);       //旋转总坐标系
        //绘制立方体
        cube.draw(gl);
    }
    public void onSurfaceChanged(GL10 gl,int width,int height) {
        gl.glViewport(0,0,width,height);            //设置 OpenGL 场景的大小
        float ratio = (float) width / height;       //计算透视视窗的宽度、高度比
        gl.glMatrixMode(GL10.GL_PROJECTION);        //将当前矩阵模式设为投影矩阵
        gl.glLoadIdentity();                        //初始化单位矩阵
        //设置透视视窗的空间大小
        GLU.gluPerspective(gl,45.0f,ratio,1,100f);
    }
}
```

(4) 打开 src\fs.gl10_3d 包下的 GLCube.java 文件，在例 7-7 的基础上为立方体应用纹理贴图。代码为：

```
package fs.gl10_3d;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.IntBuffer;
import javax.microedition.khronos.opengles.GL10;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.opengl.GLUtils;
/**
```

```
  * 定义一个简单的模型：立方体
  */
public class GLCube {
    private final IntBuffer mVertexBuffer;            //顶点坐标数据缓冲
    private IntBuffer mTextureBuffer;                 //纹理贴图数据缓冲
    public GLCube() {
        int one = 65536;
        int half = one / 2;
        int vertices[] = {
                //前面
                - half, - half,half,half, - half,half,
                - half,half,half,half,half,half,
                //背面
                - half, - half, - half, - half,half, - half,
                half, - half, - half,half,half, - half,
                //左面
                - half, - half,half, - half,half,half,
                - half, - half, - half, - half,half, - half,
                //右面
                half, - half, - half,half,half, - half,
                half, - half,half,half,half,half,
                //上面
                - half,half,half,half,half,half,
                - half,half, - half,half,half, - half,
                //下面
                - half, - half,half, - half, - half, - half,
                half, - half,half,half, - half, - half,
        }; //定义顶点位置
        //创建顶点坐标数据缓冲
        ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length * 4);
        vbb.order(ByteOrder.nativeOrder());           //设置字节顺序
        mVertexBuffer = vbb.asIntBuffer();            //转换为 int 型缓冲
        mVertexBuffer.put(vertices);                  //向缓冲中放入顶点坐标数据
        mVertexBuffer.position(0);                    //设置缓冲区的起始位置
        /** ========== 纹理贴图 ============== */
        int texCoords[] = {
                //前面
                0,one,one,one,0,0,one,0,
                //后面
                one,one,one,0,0,one,0,0,
                //左面
                one,one,one,0,0,one,0,0,
                //右面
                one,one,one,0,0,one,0,0,
                //上面
                one,0,0,0,one,one,0,one,
                //下面
                0,0,0,one,one,0,one,one,};            //定义贴图坐标数据
        ByteBuffer tbb = ByteBuffer.allocateDirect(texCoords.length * 4);
        tbb.order(ByteOrder.nativeOrder());           //设置字节顺序
        mTextureBuffer = tbb.asIntBuffer();           //转换为 int 型缓冲
        mTextureBuffer.put(texCoords);                //向缓冲中放入贴图坐标数据
        mTextureBuffer.position(0);                   //设置缓冲区的起始位置
    }
```

```
    public void draw(GL10 gl) {
gl.glVertexPointer(3,GL10.GL_FIXED,0,mVertexBuffer);    //为画笔指定顶点坐标数据
        //绘制 FRONT 和 BACK 两个面
        gl.glColor4f(1,1,1,1);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP,0,4);    //绘制图形
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP,4,4);    //绘制图形
        //绘制 LEFT 和 RIGHT 两个面
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP,8,4);    //绘制图形
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP,12,4);   //绘制图形
        //绘制 TOP 和 BOTTOM 两个面
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP,16,4);   //绘制图形
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP,20,4);   //绘制图形
        /** =============== 纹理贴图 ================== */
        gl.glTexCoordPointer(2,GL10.GL_FIXED,0,mTextureBuffer);//为画笔指定贴图坐标数据
    }
    /**
     * 进行纹理贴图
     */
    void loadTexture(GL10 gl,Context context,int resource) {
        Bitmap bmp = BitmapFactory.decodeResource(context.getResources(),
                resource);                                        //加载位图
    GLUtils.texImage2D(GL10.GL_TEXTURE_2D,0,bmp,0)               //使用图片生成纹理
        bmp.recycle();                                            //释放资源
    }
}
```

运行程序，立方体纹理贴图效果如图 7-9 所示。



图 7-9　纹理贴图

7.3.4 Android 3D旋转

到目前为止,绘制的3D物体还是静止的,为了更好地看到3D效果,还可以为其添加旋转效果。这样就可以达到动画效果了。要实现旋转比较简单,只需要使用GL10的glRotatef()方法不断地旋转要旋转的对象即可。glRotatef()方法的格式为:

```
glRotatef(float angle,float x,float y,float z)
```

其中,参数angle通常为一个变量,表示对象转过的角度;x表示X轴的旋转方向(值为1表示顺时针、−1表示逆时针方向、0表示不旋转);y表示Y轴的旋转方向(值为1表示顺时针、−1表示逆时针方向、0表示不旋转);z表示Z轴的旋转方向(值为1表示顺时针、−1表示逆时针方向、0表示不旋转)。

例如,要将对象经过Y轴旋转N角度。代码为:

```
gl.glRoatef(N,0,1,0)
```

下面通过一个实例来演示3D图形的旋转。

【例7-9】 对例7-8的绘制的立方体进行旋转。其主要实现步骤为:

(1) 打开src/fs.gl10_3d包下的Renderertest.java文件,在例7-8的基础上实现3D图形的旋转。代码为:

```
package fs.gl10_3d;
import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;
import android.content.Context;
import android.opengl.GLSurfaceView;
import android.opengl.GLU;
public class Renderertest implements GLSurfaceView.Renderer {
    private final GLCube cube;                              //立方体对象
    private Context context;
    private long startTime;                                 //定义变量保存开始时间
    public Renderertest(Context context) {
        cube = new GLCube();                                //实例化立方体对象
        this.context = context;
    startTime = System.currentTimeMillis();        //为成员变量startTime赋初始值为当前时间
    }
    public void onSurfaceCreated(GL10 gl,EGLConfig config) {
        gl.glClearColor(0.7f,0.9f,0.9f,1.0f);              //设置窗体背景颜色
        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);       //启用顶点坐标数组
        gl.glDisable(GL10.GL_DITHER);                       //关闭抗抖动
        gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,GL10.GL_FASTEST);
        //设置系统对透视进行修正
        gl.glShadeModel(GL10.GL_SMOOTH);                    //设置阴影平滑模式
        gl.glEnable(GL10.GL_DEPTH_TEST);                    //启用深度测试
        gl.glDepthFunc(GL10.GL_LEQUAL);                     //设置深度测试的类型
        /** =========== 应用纹理贴图 =============== */
gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);        //启用贴图坐标数组
        gl.glEnable(GL10.GL_TEXTURE_2D);                    //启用纹理贴图
        cube.loadTexture(gl,context,R.drawable.bj2);        //进行纹理贴图
    }
```

```
    public void onDrawFrame(GL10 gl) {
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
        //清除颜色缓存和深度缓存
        gl.glMatrixMode(GL10.GL_MODELVIEW);             //设置使用模型矩阵进行变换
        gl.glLoadIdentity();                            //初始化单位矩阵
        //当使用 GL_MODELVIEW 模式时,必须设置视点,也就是观察点
        GLU.gluLookAt(gl,0,0,-5,0f,0f,0f,0f,1.0f,0.0f);
        gl.glRotatef(1000,-0.1f,-0.1f,0.05f);          //旋转总坐标系
        /** ================ 旋转 ==================== */
        long elapsed = System.currentTimeMillis() - startTime;  //计算逝去的时间
        gl.glRotatef(elapsed * (30f / 1000f),0,1,0);            //在 Y 轴上旋转 30 度
        gl.glRotatef(elapsed * (15f / 1000f),1,0,0);            //在 X 轴上旋转 15 度
        //绘制立方体
        cube.draw(gl);
    }
    public void onSurfaceChanged(GL10 gl,int width,int height) {
        gl.glViewport(0,0,width,height);            //设置 OpenGL 场景的大小
        float ratio = (float) width / height;       //计算透视视窗的宽度、高度比
        gl.glMatrixMode(GL10.GL_PROJECTION);        //将当前矩阵模式设为投影矩阵
        gl.glLoadIdentity();                        //初始化单位矩阵
        //设置透视视窗的空间大小
        GLU.gluPerspective(gl,45.0f,ratio,1,100f);
    }
}
```

(2) 其他文件采用例 7-8 的代码。

运行程序,3D 纹理贴图旋转的效果如图 7-10 所示。

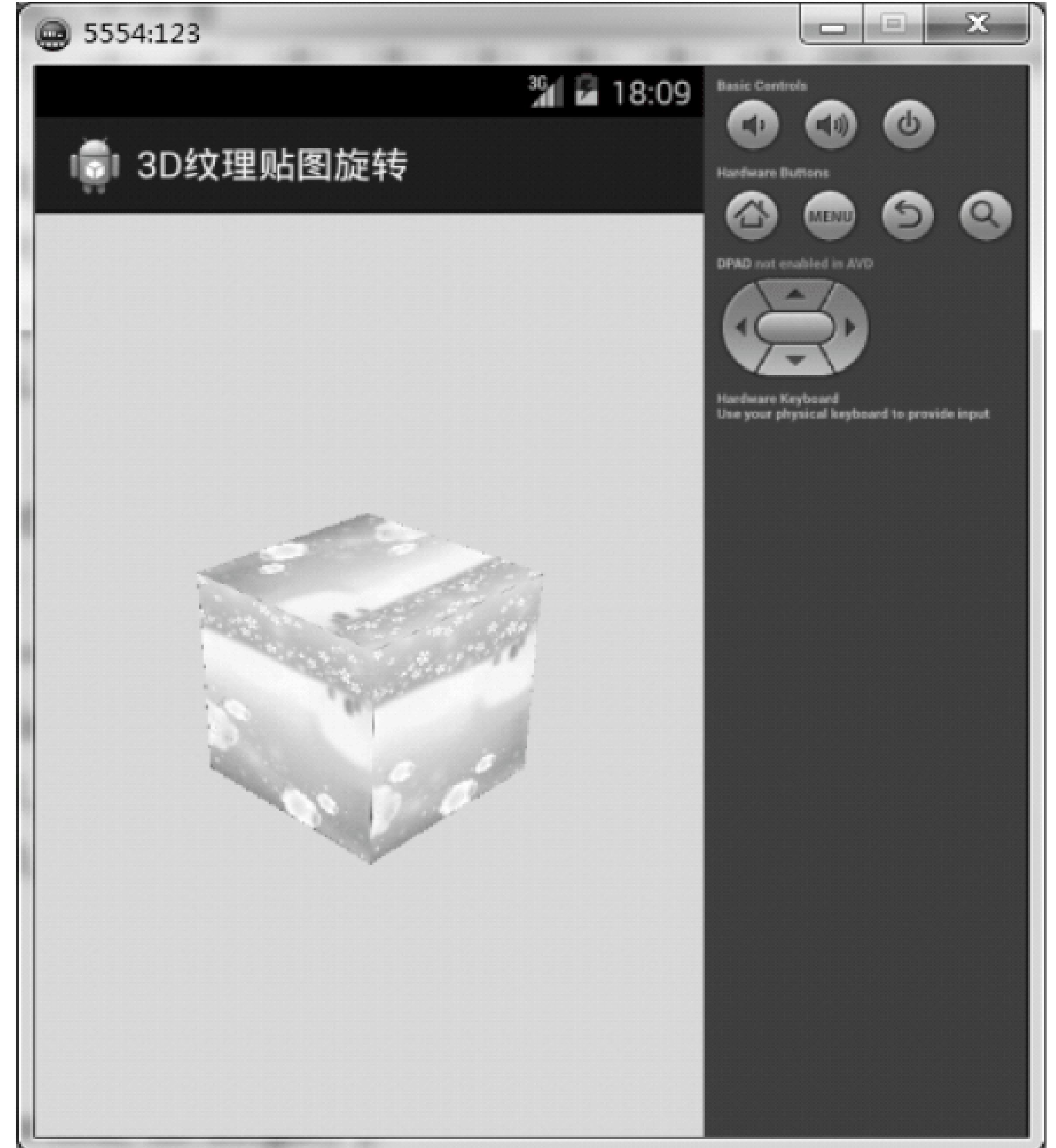


图 7-10　旋转的 3D 纹理贴图

7.3.5 Android 3D 光照

为了使程序效果更加完美、逼真，还可以让其模拟光照效果。在为物体添加光照效果前，先来了解一下 3D 图形支持的光照类型。所有的 3D 图形都支持以下 3 种光照类型。

- 环境光：一种普通的光线，光线会照亮整个场景，即使对象背对着光线也可以。
- 散射光：柔和的方向性光线。例如，荧光板上发出的光线就是这种散射光。场景中的大部分光线通常来源于散射光源。
- 镜面高光：耀眼的光线，通常来源于明亮的点光源。与有光泽的材料结合使用时，这种光会带来高光效果，增加场景的真实感。

在 OpenGL 中添加光照效果，通常分为以下两个步骤。

(1) 光线

在定义光照效果时，通常需要定义光线，即为场景添加光源。这可以通过 GL10 提供的 glLightfv()方法实现。glLightfv()方法的格式为：

```
glLightfv(int light,int pname,float[] params,int offset)
```

其中，light 表示光源的 ID，当程序中包含多个光源时，可以通过这个 ID 来区分光源；pname 表示光源的类型(参数值为 GL10.GL_AMBIENT 表示环境光，参数值为 GL10.GL_DIFFUSE 表示散射光)；params 表示光源数组；offset 表示偏移量。

例如，要定义一个发出白色的全方向的光源，可用以下代码：

```
float lightAmbient[] = new float[]{0.2f,0.2f,0.2f,1};          //定义环境光
float lightDiffuse[] = new float[]{1,1,1};                     //定义散射光
float lightPos[] = new float[]{1,1,1,1};                       //定义光源的位置
gl.glEnable(GL10.GL_LIGHTING);                                 //启用光源
gl.glEnable(GL10.GL_L1GHT10);                                  //启动 0 号光源
gl.glEnable(GL10.GL_L1GHT10,GL10.GL_AMBIENT,lightAmbient,0);   //设置环境光
gl.glEnable(GL10.GL_L1GHT10,GL10.GL_DIFFUSE,lightDiffuse,0);   //设置散射光
gl.glEnable(GL10.GL_L1GHT10,GL10.GL_POSITION,lightPos,0);      //设置光源的位置
```

注意：在定义和设置淘汰后，还需要使用 glEnable()方法启动光源，否则，设置的光源将不起作用。

(2) 被照射的物体

在定义光照效果时，通常需要定义被照射物体的制作材料，因为不同材料的光线反射情况是不同的。使用 GL10 提供的 glMaterialfv()方法可以设置材质的环境光和散射光。glMaterialfv()方法的格式为：

```
glMaterialfv(int face,int pname,float[] params,int offset)
```

其中，face 为正面还是背面材质设置光源；pname 为光源的类型(参数值 GL10.GL_AMBIENT 表示环境光，参数值为 GL10.GL_DIFFUSE 表示散射光)；params 表示光源数组；offset 表示偏移量。

例如，定义一个不是很高的纸质的物体，可使用以下代码：

```
float matAmbient[] = new float[]{1,1,1};                       //定义材质的环境光
float matDiffuse[] = float[]{1,1,1};                           //定义材质的散射光
```

```
//设置材质的环境光
gl.glMaterialfv(GL10.GL_FRONT_AND_BACK,GL10.GL_AMBIENT,matAmbient,0);
//设置材质的散射光
gl.glMaterialfv(GL10.GL_FRONT_AND_BACK,GL10.GL_DIFFUSE,matDiffuse,0);
```

下面通过一个实例来演示为立方体添加光照效果。

【例 7-10】 在例 7-9 绘制的旋转立方体上添加光照效果。其具体实现步骤为：

(1) 打开 src/fs.gl10_3d 包下的 Renderertest.java 文件，在例 7-9 的基础上实现为旋转的立方体添加光照效果。代码为：

```
package fs.gl10_3d;
import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;
import android.content.Context;
import android.opengl.GLSurfaceView;
import android.opengl.GLU;
public class Renderertest implements GLSurfaceView.Renderer {
    private final GLCube cube;                              //立方体对象
    private Context context;
    private long startTime;                                 //定义变量保存开始时间
    public Renderertest(Context context) {
        cube = new GLCube();                                //实例化立方体对象
        this.context = context;
    startTime = System.currentTimeMillis();
    //为成员变量 startTime 赋初始值为当前时间
    }
    public void onSurfaceCreated(GL10 gl,EGLConfig config) {
        gl.glClearColor(0.7f,0.9f,0.9f,1.0f);              //设置窗体背景颜色
        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);       //启用顶点坐标数组
        gl.glDisable(GL10.GL_DITHER);                       //关闭抗抖动
        gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,GL10.GL_FASTEST);
                                                            //设置系统对透视进行修正
        gl.glShadeModel(GL10.GL_SMOOTH);                    //设置阴影平滑模式
        gl.glEnable(GL10.GL_DEPTH_TEST);                    //启用深度测试
        gl.glDepthFunc(GL10.GL_LEQUAL);                     //设置深度测试的类型
        /** ============ 应用纹理贴图 =============== */
gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);        //启用贴图坐标数组
        gl.glEnable(GL10.GL_TEXTURE_2D);                    //启用纹理贴图
        cube.loadTexture(gl,context,R.drawable.bj2);        //进行纹理贴图
        /** ================= 光照效果 ===================== */
        //为物体添加环境光和散射光
        float matAmbient[] = new float[]{1,1,1,1};          //定义材质的环境光
        float matDiffuse[] = new float[]{1,1,1,1};          //定义材质的散射光
    //设置材质的环境光
gl.glMaterialfv(GL10.GL_FRONT_AND_BACK,GL10.GL_AMBIENT,matAmbient,0);
    //设置材质的散射光
gl.glMaterialfv(GL10.GL_FRONT_AND_BACK,GL10.GL_DIFFUSE,matDiffuse,0);
        //添加光线
        float lightAmbient[] = new float[]{0.2f,0.2f,0.2f,1}; //定义环境光
        float lightDiffuse[] = new float[]{1,1,1,1};        //定义散射光
        float lightPos[] = new float[]{1,1,1,1};            //定义光源的位置
        gl.glEnable(GL10.GL_LIGHTING);                      //启用光源
        gl.glEnable(GL10.GL_LIGHT0);                        //启用 0 号光源
```

```
                                                    //设置环境光
        gl.glLightfv(GL10.GL_LIGHT0,GL10.GL_AMBIENT,lightAmbient,0);
                                                    //设置散射光
       gl.glLightfv(GL10.GL_LIGHT0,GL10.GL_DIFFUSE,lightDiffuse,0);
                                                    //设置光源的位置
        gl.glLightfv(GL10.GL_LIGHT0,GL10.GL_POSITION,lightPos,0);    }
    public void onDrawFrame(GL10 gl) {
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
                                                    //清除颜色缓存和深度缓存
        gl.glMatrixMode(GL10.GL_MODELVIEW);         //设置使用模型矩阵进行变换
        gl.glLoadIdentity();                        //初始化单位矩阵
        //当使用 GL_MODELVIEW 模式时,必须设置视点,也就是观察点
        GLU.gluLookAt(gl,0,0,-5,0f,0f,0f,0f,1.0f,0.0f);
        gl.glRotatef(1000,-0.1f,-0.1f,0.05f);       //旋转总坐标系
        /** ================ 旋转 ==================== */
        long elapsed = System.currentTimeMillis() - startTime;  //计算逝去的时间
        gl.glRotatef(elapsed * (30f / 1000f),0,1,0);            //在 Y 轴上旋转 30 度
        gl.glRotatef(elapsed * (15f / 1000f),1,0,0);            //在 X 轴上旋转 15 度
        //绘制立方体
        cube.draw(gl);
    }
    public void onSurfaceChanged(GL10 gl,int width,int height) {
        gl.glViewport(0,0,width,height);                  //设置 OpenGL 场景的大小
        float ratio = (float) width / height;       //计算透视视窗的宽度、高度比
        gl.glMatrixMode(GL10.GL_PROJECTION);        //将当前矩阵模式设为投影矩阵
        gl.glLoadIdentity();                        //初始化单位矩阵
        //设置透视视窗的空间大小
        GLU.gluPerspective(gl,45.0f,ratio,1,100f);
    }
}
```

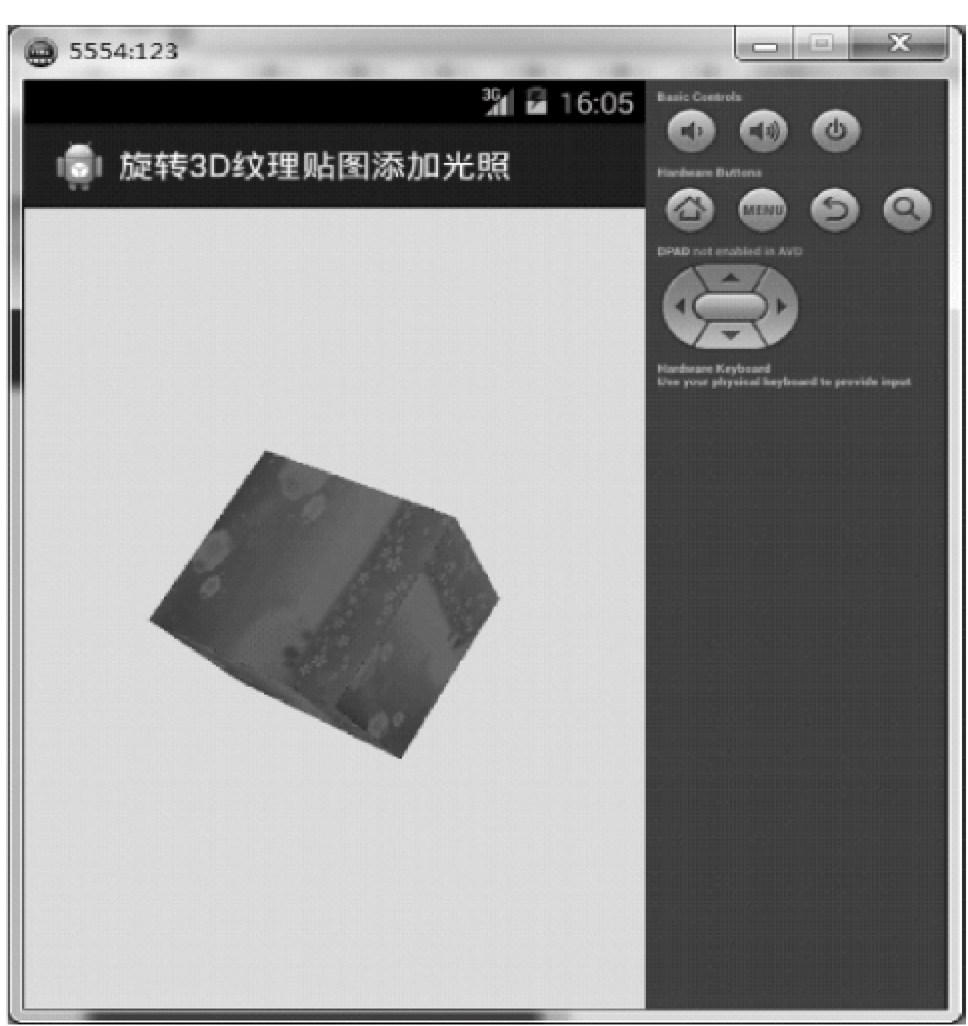
(2) 其他程序采用默认代码。

运行程序,效果如图 7-11(a)及(b)所示的变换过程。





(a) 旋转的立方体



(b) 添加光照的旋转立方体

图 7-11　为立方体添加光照效果

7.3.6 Android 3D 透明度

在游戏中，经常需要应用透明效果，使用 OpenGL ES 实现简单的透明效果也比较简单，只需要应用以下代码：

```
gl.glDisable(GL10.GL_DEPTH_TEST);                //关闭深度测试
gl.glEnable(GL10.GL_BLEND);                      //打开混合
//使用 alpha 通道值进行混色，从而达到透明效果
gl.glBlendFunc(GL10.GL_SRC_ALPHA,GL10.GL_ONE);
```

说明：实现透明效果时，需要关闭深度测试，并且打开混合效果，然后才能使用 GL10 类的 glBlendFunc()方法进行混色，从而达到透明效果。

下面通过一个实例来演示透明效果的使用。

【例 7-11】 在例 7-10 的基础上创建一个透明的、不断旋转的立方体。其具体实现步骤为：

(1) 打开 src/fs.gl10_3d 包下的 Renderertest.java 文件，在例 7-10 的基础上实现为旋转的立方体添加光照效果。代码为：

```
package fs.gl10_3d;
import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;

import android.content.Context;
import android.opengl.GLSurfaceView;
import android.opengl.GLU;
public class Renderertest implements GLSurfaceView.Renderer {
    private final GLCube cube;                              //立方体对象
    private Context context;
    private long startTime;                                 //定义变量保存开始时间
    public Renderertest(Context context) {
        cube = new GLCube();                                //实例化立方体对象
        this.context = context;
    startTime = System.currentTimeMillis();         //为成员变量 startTime 赋初始值为当前时间
    }
    public void onSurfaceCreated(GL10 gl,EGLConfig config) {
        gl.glClearColor(0.7f,0.9f,0.9f,1.0f);               //设置窗体背景颜色
        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);       //启用顶点坐标数组
        gl.glDisable(GL10.GL_DITHER);                       //关闭抗抖动
        gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,GL10.GL_FASTEST);
                                                            //设置系统对透视进行修正
        gl.glShadeModel(GL10.GL_SMOOTH);                    //设置阴影平滑模式
        gl.glEnable(GL10.GL_DEPTH_TEST);                    //启用深度测试
        gl.glDepthFunc(GL10.GL_LEQUAL);                     //设置深度测试的类型
        /** ================= 透明效果 ====================== */
        gl.glDisable(GL10.GL_DEPTH_TEST);                   //关闭深度测试
        gl.glEnable(GL10.GL_BLEND);                         //打开混合
        //使用 alpha 通道值进行混色，从而达到透明效果
        gl.glBlendFunc(GL10.GL_SRC_ALPHA,GL10.GL_ONE);
        /** ============ 应用纹理贴图 =============== */
```

```
gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);        //启用贴图坐标数组
        gl.glEnable(GL10.GL_TEXTURE_2D);                     //启用纹理贴图
        cube.loadTexture(gl,context,R.drawable.bj2);         //进行纹理贴图
        /** ================= 光照效果 ===================== */
        //为物体添加环境光和散射光
        float matAmbient[] = new float[]{1,1,1,1};           //定义材质的环境光
        float matDiffuse[] = new float[]{1,1,1,1};           //定义材质的散射光
     //设置材质的环境光
gl.glMaterialfv(GL10.GL_FRONT_AND_BACK,GL10.GL_AMBIENT,matAmbient,0);
    //设置材质的散射光
gl.glMaterialfv(GL10.GL_FRONT_AND_BACK,GL10.GL_DIFFUSE,matDiffuse,0);
        //添加光线
        float lightAmbient[] = new float[]{0.2f,0.2f,0.2f,1};//定义环境光
        float lightDiffuse[] = new float[]{1,1,1,1};         //定义散射光
        float lightPos[] = new float[]{1,1,1,1};             //定义光源的位置
        gl.glEnable(GL10.GL_LIGHTING);                       //启用光源
        gl.glEnable(GL10.GL_LIGHT0);                         //启用 0 号光源
       //设置环境光
        gl.glLightfv(GL10.GL_LIGHT0,GL10.GL_AMBIENT,lightAmbient,0);
    //设置散射光
    gl.glLightfv(GL10.GL_LIGHT0,GL10.GL_DIFFUSE,lightDiffuse,0);
            gl.glLightfv(GL10.GL_LIGHT0,GL10.GL_POSITION,lightPos,0);  //设置光源的位置
    }
    public void onDrawFrame(GL10 gl) {
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
                                                             //清除颜色缓存和深度缓存
        gl.glMatrixMode(GL10.GL_MODELVIEW);                  //设置使用模型矩阵进行变换
        gl.glLoadIdentity();                                 //初始化单位矩阵
        //当使用 GL_MODELVIEW 模式时,必须设置视点,也就是观察点
        GLU.gluLookAt(gl,0,0,-5,0f,0f,0f,0f,1.0f,0.0f);
        gl.glRotatef(1000,-0.1f,-0.1f,0.05f);                //旋转总坐标系
        /** ================ 旋转 ===================== */
        long elapsed = System.currentTimeMillis() - startTime; //计算逝去的时间
        gl.glRotatef(elapsed * (30f / 1000f),0,1,0);           //在 y 轴上旋转 30 度
        gl.glRotatef(elapsed * (15f / 1000f),1,0,0);           //在 x 轴上旋转 15 度
        //绘制立方体
        cube.draw(gl);
    }
    public void onSurfaceChanged(GL10 gl,int width,int height) {
        gl.glViewport(0,0,width,height);               //设置 OpenGL 场景的大小
        float ratio = (float) width / height;          //计算透视视窗的宽度、高度比
        gl.glMatrixMode(GL10.GL_PROJECTION);           //将当前矩阵模式设为投影矩阵
        gl.glLoadIdentity();                           //初始化单位矩阵
        //设置透视视窗的空间大小
        GLU.gluPerspective(gl,45.0f,ratio,1,100f);
    }
}
```

(2) 其他程序代码采用默认形式。

运行程序,得到透明且旋转的立方体如图 7-12 所示。



图 7-12　透明且旋转的立方体

第 8 章 Android 数据存储与共享

Android 为开发人员提供了多种持久化应用数据的方式，具体选择哪种方式需要具体问题具体分析，例如数据是否仅限于本程序使用，还是可以用于其他程序以及保存数据所占用的空间等。在 Android 中主要提供了 4 种数据存储技术，分别为 SharedPreferences、Files、SQLite 及 NetWork 数据库。

8.1 SharedPreferences 存储数据

SharedPreferences 存储方式适用于简单数据的保存，如配置属性、保存用户名等具有配置性质的数据保存，但是不适合数据比较大的保存方式。一般在 Activity 中重载窗口状态 onSaveInstanceState 保存一般使用 SharedPreferences 完成，它提供了 Android 平台常规的长整型(Long)、整型(Int)、字符串型(String)的保存。

SharedPreferences 类似过去 Windows 系统上的 ini 配置文件，但是它分为多种权限，可以全局共享访问，Android 最终是以 xml 方式来保存(在 Android 系统中，这些信息以 XML 文件的形式保存在/data/data/PACKAGE_NAME /shared_prefs 目录下)，整体效率来看不是特别的高，对于常规的轻量级而言比 SQLite 要好很多，如果真的存储量不大可以考虑自己定义文件格式。xml 处理时 Dalvik 会通过自带底层的本地 XML Parser 解析，例如 XMLpull 方式，这样对于内存资源占用比较好。

在 Android 中有两种方式可以获得 SharedPreferences 对象。

- getSharedPreferences()：如果需要多个使用名称来区分共享文件，则可以使用该方法，其第一个参数就是共享文件的名称。对于使用同一个名称获得的多个 SharedPreferences 引用，其指向同一个对象。
- getPreferences()：如果 Activity 仅需要一个共享文件，则可以使用该方法。因为只有一个文件，它并不需要提供名称。

完成向 SharedPreferences 类中增加值的步骤为：

(1) 调用 SharedPreferences 类的 edit()方法获得 SharedPreferences. Editor 对象。

(2) 调用诸如 putBoolean()、putString()等方法增加值。

(3) 使用 commit()方法提交新值。

从 SharedPreferences 类中读取值时，主要使用该类中定义的 getXXX()方法。

下面通过一个实例来演示 SharedPreferences 存储数据的用法。

【例 8-1】 利用 SharedPreferences 类实现一个界面，并在界面输入数据进行保存。其具体实现步骤为：

（1）在 Eclipse 中创建一个 Android 应用项目，命名为 SharedPreferences_test。

（2）打开 res\layout 目录下的 main.xml 布局文件，在文件中增加文本框、编辑框等控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
    <RelativeLayout
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content">
        <TextView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "姓名"
            android:textSize = "30px"
            android:id = "@ + id/nameLable" />
        <EditText
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_toRightOf = "@id/nameLable"
            android:layout_alignTop = "@id/nameLable"
            android:layout_marginLeft = "10px"
            android:id = "@ + id/name" />
    </RelativeLayout>
    <RelativeLayout
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content">
        <TextView android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:textSize = "30px"
            android:text = "年龄"
            android:id = "@ + id/ageLable" />
        <EditText
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_toRightOf = "@id/ageLable"
            android:layout_alignTop = "@id/ageLable"
            android:layout_marginLeft = "10px"
            android:id = "@ + id/age" />
    </RelativeLayout>
    <RelativeLayout
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content">
        <Button
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "保存数据"
```

```
                android:id = "@ + id/button" />
            < Button
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:text = "显示数据"
                android:layout_toRightOf = "@id/button"
                android:layout_alignTop = "@id/button"
                android:id = "@ + id/showButton" />
        </RelativeLayout >
        < TextView android:layout_width = "fill_parent"
                android:layout_height = "wrap_content"
                android:textSize = "20px"
                android:id = "@ + id/showText" />
</LinearLayout >
```

(3) 打开 src\fs. sharedpreferences_test 包下的 MainActivity. java 文件,在文件中获取用户的姓名及年龄,当单击“保存数据”按钮时,实现 SharedPreferences 保存数据的功能,当单击界面中的“显示数据”按钮时,显示所输入的姓名及年龄。代码为:

```
package fs. sharedpreferences_test;
import android. app. Activity;
import android. content. Context;
import android. content. SharedPreferences;
import android. content. SharedPreferences. Editor;
import android. os. Bundle;
import android. view. View;
import android. widget. Button;
import android. widget. EditText;
import android. widget. TextView;
import android. widget. Toast;
public class MainActivity extends Activity {
    private EditText nameText;
    private EditText ageText;
    private TextView resultText;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super. onCreate(savedInstanceState);
        setContentView(R. layout. main);
        nameText  =  (EditText)this. findViewById(R. id. name);
        ageText  =  (EditText)this. findViewById(R. id. age);
        resultText  =  (TextView)this. findViewById(R. id. showText);
        Button button  =  (Button)this. findViewById(R. id. button);
        Button showButton  =  (Button)this. findViewById(R. id. showButton);
        button. setOnClickListener(listener);
        showButton. setOnClickListener(listener);
        //回显
        SharedPreferences sharedPreferences = getSharedPreferences("ljq123",
                Context. MODE_WORLD_READABLE + Context. MODE_WORLD_WRITEABLE);
        String nameValue  =  sharedPreferences. getString("name","");
        int ageValue  =  sharedPreferences. getInt("age",1);
        nameText. setText(nameValue);
```

```java
            ageText.setText(String.valueOf(ageValue));
        }
        private View.OnClickListener listener = new View.OnClickListener(){
            public void onClick(View v) {
                Button button = (Button)v;
                //ljq123 文件存放在/data/data/<package name>/shared_prefs 目录下
                SharedPreferences sharedPreferences = getSharedPreferences("ljq123",
                        Context.MODE_WORLD_READABLE + Context.MODE_WORLD_WRITEABLE);
                switch (button.getId()) {
                case R.id.button:
                    String name = nameText.getText().toString();
                    int age = Integer.parseInt(ageText.getText().toString());
                    Editor editor = sharedPreferences.edit(); //获取编辑器
                    editor.putString("name",name);
                    editor.putInt("age",age);
                    editor.commit();                            //提交修改
          Toast.makeText(MainActivity.this,"保存成功",Toast.LENGTH_LONG).show();
                    break;
                case R.id.showButton:
                    String nameValue = sharedPreferences.getString("name","");
                    int ageValue = sharedPreferences.getInt("age",1);
                    resultText.setText("姓名:" + nameValue + ",年龄:" + ageValue);
                    break;
                }
            }
        };
    }
```

运行程序,默认效果如图 8-1(a)所示,当在界面中输入对应的姓名、年龄后单击界面中的“保存数据”按钮,再单击“显示数据”按钮,效果如图 8-1(b)所示。



(a) 默认界面　　　　(b) 显示保存数据

图 8-1　SharedPreferences 保存数据

在例 8-1 中,演示了怎样使用私有的 SharedPreferences 来实现保存数据。除了 MODE_PRIVATE(默认模式),还有另外两种模式:MODE_WORLD_READABLE 和 MODE_

WORLD_WRITEABLE。它们分别表示对其他应用程序是否可读与可写。下面演示这两个模式的使用。

【例 8-2】 利用 SharedPreferences 实现在两个不同的 Android 项目中传递数据。其具体实现步骤为：

(1) 在 Eclipse 中创建两个 Android 应用项目，分别命名为 SharedPreferences_1 和 SharedPreferences_2。

(2) 打开 SharedPreferences_1 项目中的 res\layout 目录下的 main. xml 布局文件，在文件中添加文本框、编辑框、按钮等控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc"
    android:orientation = "vertical" >
    <LinearLayout
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content" >
        <TextView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "全局可读"
            android:textColor = "@android:color/white"
            android:textSize = "30dp" />
        <EditText
            android:id = "@ + id/worldRead"
            android:layout_width = "0dip"
            android:layout_height = "wrap_content"
            android:layout_weight = "1"
            android:inputType = "text"
            android:textColor = "@android:color/white"
            android:textSize = "25dp" >
            <requestFocus />
        </EditText>
    </LinearLayout>
    <LinearLayout
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content" >
        <TextView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "全局可写"
            android:textColor = "@android:color/white"
            android:textSize = "30dp" />
        <EditText
            android:id = "@ + id/worldWrite"
            android:layout_width = "0dip"
            android:layout_height = "wrap_content"
            android:layout_weight = "1"
```

```
            android:inputType = "text"
            android:textColor = "@android:color/white"
            android:textSize = "25dp" />
    </LinearLayout>
    <LinearLayout
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content" >
        <TextView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "全局可读可写"
            android:textColor = "@android:color/white"
            android:textSize = "30dp" />
        <EditText
            android:id = "@ + id/worldReadWrite"
            android:layout_width = "0dip"
            android:layout_height = "wrap_content"
            android:layout_weight = "1"
            android:inputType = "text"
            android:textColor = "@android:color/white"
            android:textSize = "25dp" />
    </LinearLayout>
    <Button
        android:id = "@ + id/save"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "保存"
        android:textColor = "@android:color/white"
        android:textSize = "25dp" />
</LinearLayout>
```

(3) 打开 src\fs. sharedpreferences_1 包下的 MainActivity. java 文件，在文件中创建 3 个名称和权限都不相同的 SharedPreferences，向其中写入用户所需要保存的值。代码为：

```
package fs.sharedpreferences_1;
import android.app.Activity;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
public class MainActivity extends Activity {
    /** 第 1 次调用 activity 活动 */
    private EditText worldReadET;
    private EditText worldWriteET;
    private EditText worldReadWriteET;
    private SharedPreferences worldReadSP;
    private SharedPreferences worldWriteSP;
    private SharedPreferences worldReadWriteSP;
    @Override
```

```
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);                      //调用父类方法
            setContentView(R.layout.main);                           //应用自定义布局文件
            worldReadET = (EditText) findViewById(R.id.worldRead);   //获得全局可读控件
            worldWriteET = (EditText) findViewById(R.id.worldWrite); //获得全局可写控件
             //获得全局可读可写控件
            worldReadWriteET = (EditText) findViewById(R.id.worldReadWrite);
       worldReadSP = getSharedPreferences("worldRead",MODE_WORLD_READABLE);
     worldWriteSP = getSharedPreferences("worldWrite",MODE_WORLD_WRITEABLE);
           worldReadWriteSP = getSharedPreferences("worldReadWrite",MODE_WORLD_READABLE +
    MODE_WORLD_WRITEABLE);
            Button save = (Button) findViewById(R.id.save);
            save.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    String worldReadS = worldReadET.getText().toString();
                    String worldWriteS = worldWriteET.getText().toString();
                    String worldReadWriteS = worldReadWriteET.getText().toString();
                    Editor worldReadE = worldReadSP.edit();
                    Editor worldWriteE = worldWriteSP.edit();
                    Editor worldReadWriteE = worldReadWriteSP.edit();
                    worldReadE.putString("key",worldReadS);
                    worldWriteE.putString("key",worldWriteS);
                    worldReadWriteE.putString("key",worldReadWriteS);
                    worldReadE.commit();
                    worldWriteE.commit();
                    worldReadWriteE.commit();
                }
            });
        }
    }
```

(4) 打开 SharedPreferences_2 项目中的 res\layout 目录下的 main.xml 布局文件，在文件中添加 3 个文本框控件，用于显示 SharedPreferences_1 中的数据。代码为：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#000000"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/worldRead"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@android:color/white"
        android:textSize="30dp" />
    <TextView
        android:id="@+id/worldWrite"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
            android:textColor = "@android:color/white"
            android:textSize = "30dp" />
        < TextView
            android:id = "@ + id/worldReadWrite"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:textColor = "@android:color/white"
            android:textSize = "30dp" />
</LinearLayout >
```

(5) 打开 src\fs. sharedpreferences_2 包下的 MainActivity. java 文件，在该文件中，获得在项目 SharedPreferences_1 中定义的 SharedPreferences，然后显示其值。代码为：

```
package fs. sharedpreferences_2;
import android. app. Activity;
import android. content. Context;
import android. content. SharedPreferences;
import android. content. pm. PackageManager. NameNotFoundException;
import android. os. Bundle;
import android. widget. TextView;
public class MainActivity extends Activity {
    private SharedPreferences worldReadSP;
    private SharedPreferences worldWriteSP;
    private SharedPreferences worldReadWriteSP;
    private TextView worldReadTV;
    private TextView worldWriteTV;
    private TextView worldReadWriteTV;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super. onCreate(savedInstanceState);
        setContentView(R. layout. main);
        Context otherContext = null;
        try {
      otherContext = createPackageContext("fs. sharedpreferences_1",MODE_PRIVATE);
        } catch (NameNotFoundException e) {
            e. printStackTrace();
        }

        worldReadSP = otherContext. getSharedPreferences("worldRead",MODE_WORLD_READABLE);
        worldWriteSP = otherContext. getSharedPreferences("worldWrite",MODE_WORLD_WRITEABLE);
        worldReadWriteSP = otherContext. getSharedPreferences("worldReadWrite",MODE_WORLD_
READABLE + MODE_WORLD_WRITEABLE);
        worldReadTV = (TextView) findViewById(R. id. worldRead);
        worldWriteTV = (TextView) findViewById(R. id. worldWrite);
        worldReadWriteTV = (TextView) findViewById(R. id. worldReadWrite);
        worldReadTV. setText("全局可读：" + worldReadSP. getString("key","null"));
        worldWriteTV. setText("全局可写：" + worldWriteSP. getString("key","null"));
        worldReadWriteTV. setText("全局可读可写：" + worldReadWriteSP. getString("key","null"));
    }
}
```

运行程序 SharedPreferences_1 项目，并输入对应的值，即显示如图 8-2(a)所示的接收用户信息界面，单击界面中的“保存”按钮。运行 SharedPreferences_2 项目，显示如图(b)所示的界面，界面上显示了用户刚刚输入信息的获取情况。

(a) 接收用户信息



(b) 获取显示信息

图 8-2　不同项目间的数据传递

8.2　File 存储数据

File 存储方式是较常使用的一种保存数据方式，可以保存圈套的数据。而且文件存储不仅能把数据存储在系统中也能将数据保存到 SDcard 中。

在 Android 中，使用 Files 对象存储数据主要有两种方式，一种是 Java 提供的 IO 流体系，即使用 FileOutputStream 类提供的 openFileOutput()方法和 FileInputStream 类提供的 openFileInput()方法访问磁盘上的内容文件；另一种是使用 Environment 类的 getExternalStorageDirectory()方法对 Android 模拟器的 SD 卡进行数据读写。

8.2.1　openFileOutput、openFileInput 读/写文件

使用 Java 提供的 IO 流体系可以很方便地对 Android 模拟器本地存储的数据进行读写操作，其中，FileOutputStream 类的 openFileOutput()方法用来打开相应的输出流；而 FileInputStream 类的 openFileInput()方法用来打开相应的输入流。默认情况下，使用 IO 流保存的文件仅对当前应用程序可见，对于其他应用程序(包括用户)是不可见的(即不能访问其中的数据)。如果用户卸载了该应用程序，则保存数据的文件也会一起被删除。

下面通过一个实例来演示怎样使用 Java 提供的 IO 流体系对 Android 程序中的本地文件进行操作。

【例 8-3】 使用 openFileOutput、openFileInput 方法读写文件。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 OpenFile_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中添加两个 Button 控件。

代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
        android:orientation = "vertical"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:background = "#aabbcc">
    <Button
        android:text = "创建文件"
        android:id = "@ + id/button1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"></Button>
    <Button
        android:text = "打印文件"
        android:id = "@ + id/button2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"></Button>
</LinearLayout>
```

（3）打开 src\fs.openfile_test 包下的 MainActivity.java 文件，在文件中自定义文件，并保存打印。代码为：

```
package fs.openfile_test;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
public class MainActivity extends Activity {
    private Button button1;
    private Button button2;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        button1 = (Button) this.findViewById(R.id.button1);
        button2 = (Button) this.findViewById(R.id.button2);
        button1.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
                try {
                    FileOutputStream fosRef = MainActivity.this.openFileOutput(
                            "ghab.txt",Context.MODE_PRIVATE);
                    fosRef.write("Android 精要".getBytes());
```

```
                fosRef.close();
                StringBuffer sbRef = new StringBuffer();
            FileInputStream fisRef = MainActivity.this.openFileInput("ghy.txt");
                InputStreamReader isrRef = new InputStreamReader(fisRef);
                char[] charArray = new char[2];
                int readLength = isrRef.read(charArray);
                while (readLength != -1) {
                    sbRef.append(charArray,0,readLength);
                    readLength = isrRef.read(charArray);
                }
                Log.v("读入的值: ",new String(sbRef.toString()));
                fisRef.close();
                isrRef.close();
            } catch (FileNotFoundException e) {
                //TODO 自动存根法
                e.printStackTrace();
            } catch (IOException e) {
                //TODO 自动存根法
                e.printStackTrace();
            }
        }
    });
    button2.setOnClickListener(new OnClickListener() {
        public void onClick(View arg0) {
            try {
                FileOutputStream fosRef = MainActivity.this.openFileOutput(
                        "ghy.txt",Context.MODE_APPEND);
                fosRef.write(" 我不是中国人我是追加的".getBytes());
                fosRef.close();
                StringBuffer sbRef = new StringBuffer();
            FileInputStream fisRef = MainActivity.this.openFileInput("ghy.txt");
                InputStreamReader isrRef = new InputStreamReader(fisRef);
                char[] charArray = new char[2];
                int readLength = isrRef.read(charArray);
                while (readLength != -1) {
                    sbRef.append(charArray,0,readLength);
                    readLength = isrRef.read(charArray);
                }
                Log.v("读入的值: ",new String(sbRef.toString()));
                fisRef.close();
                isrRef.close();
            } catch (FileNotFoundException e) {
                //TODO 自动存根法
                e.printStackTrace();
            } catch (IOException e) {
                //TODO 自动存根法
                e.printStackTrace();
            }
        }
    });
  }
}
```

运行程序,效果如图 8-3 所示。



图 8-3　读写文件

8.2.2　SD 卡读/写文件

每个 Android 设备都支持共享的外部存储用来保存文件,这可以是 SD 卡等可以移除的存储介质,也可以是手机内存等不可以移除的存储介质。保存的外部存储的文件都是全局可读的,而且在用户使用 USB 连接计算机后,可以修改这些文件。在 Android 程序中,对 SD 卡等外部存储的文件进行操作时,需要使用 Enironment 类的 getExternalStorageDirectory()方法,该方法用来获取外部存储器(SD 卡)的目录。

下面通过一个实例来演示 SD 卡读/写文件。

【例 8-4】　单击"根目录"按钮查询系统的文件名称并显示。在名称界面单击文件名称,如果存在子目录则进入子目录,否则弹出修改文件名称对话框。在对话框中输入将要修改的名称,单击"确定"按钮即可改变文件的名称。单击"上翻"按钮,将跳转到本界面的上一界面。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 SD_test。

(2) 打开 res\layout 目录下的 main. xml 文件,在文件中布局两个线性布局、1 个帧布局、两个 Button 控件。代码为:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
```

```
        android:paddingTop = "@dimen/activity_vertical_margin"
        tools:context = ".MainActivity"
        android:background = "@drawable/bj1">
    <LinearLayout
            android:layout_width = "fill_parent"
            android:layout_height = "fill_parent"
            android:gravity = "bottom|right">
            <Button
                android:id = "@+id/bgml"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:text = "根目录"
                android:textSize = "18dip"/>
        </LinearLayout>
        <LinearLayout
            android:layout_width = "fill_parent"
            android:layout_height = "fill_parent"
            android:gravity = "bottom|left">
            <Button
                android:id = "@+id/bsf"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:text = "上翻"
                android:textSize = "18dip"/>
        </LinearLayout>
        <LinearLayout
                android:layout_width = "fill_parent"
            android:layout_height = "360dip"
            android:orientation = "vertical">
            <LinearLayout
                android:layout_width = "fill_parent"
                android:layout_height = "wrap_content">
                <TextView
                    android:layout_width = "wrap_content"
                    android:layout_height = "wrap_content"
                    android:text = "文件列表:"
                    android:textSize = "18dip"
                    android:textColor = "#000000"/>
            </LinearLayout>
            <LinearLayout
                android:layout_width = "fill_parent"
                android:layout_height = "wrap_content"
                android:paddingTop = "10dip">
                <ListView
                    android:id = "@+id/lvwjlb"
                    android:layout_width = "fill_parent"
                    android:layout_height = "fill_parent"
                    android:textSize = "18dip"
                    android:textColor = "#000000"
                    android:divider = "#EDAB4A"/>
            </LinearLayout>
```

```
    </LinearLayout>
</RelativeLayout>
```

(3) 在 res\layout 目录下创建一个名为 dialog.xml 的文件，在文件中设置 3 个线性布局、一个 TextView、一个 EditText 及两个 Button。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/bj1">
    <LinearLayout
        android:orientation = "vertical"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:paddingLeft = "10dip"
        android:paddingRight = "10dip"
        android:paddingTop = "10dip"
        android:paddingBottom = "10dip"
        android:gravity = "center">
        <LinearLayout
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content">
            <TextView
                android:layout_width = "fill_parent"
                android:layout_height = "wrap_content"
                android:textSize = "18dip"
                android:textColor = "#ffffff"
                android:paddingLeft = "10dip"
                android:text = "请输入新的文件名称: "/>
        </LinearLayout>
        <LinearLayout
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:paddingLeft = "5dip"
            android:paddingRight = "5dip"
            android:paddingTop = "5dip">
            <EditText
                android:text = ""
                android:textSize = "18dip"
                android:textColor = "#000000"
                android:id = "@ + id/et"
                android:layout_width = "fill_parent"
                android:layout_height = "wrap_content"
                android:singleLine = "true"/>
        </LinearLayout>
        <LinearLayout
            android:orientation = "horizontal"
            android:layout_width = "fill_parent"
            android:layout_height = "fill_parent"
```

```
                android:gravity = "center">
                 < Button
                    android:text = "确定"
                    android:id = "@ + id/bOk"
                    android:layout_width = "75dip"
                    android:layout_height = "40dip"
                    android:textSize = "18dip"
                    android:gravity = "center"/>
                < Button
                    android:text = "取消"
                    android:id = "@ + id/bCancle"
                    android:layout_width = "75dip"
                    android:layout_height = "40dip"
                    android:textSize = "18dip"
                    android:gravity = "center"/>
            </LinearLayout >
        </LinearLayout >
    </LinearLayout >
```

（4）打开 src\fs.sd_test 包下的 MainActivity.java 文件，在文件中实现手机文件的修改。代码为：

```
public class MainActivity extends Activity
{
    String currentPath;                       //记录当前文件列表的父路径
    String rootPath = "/";                    //根目录
    String leavePath;                         //叶子文件
    Dialog gmDialog;                          //声明改名对话框
    ListView lv;                              //ListView 控件对象声明
    @Override
    public Dialog onCreateDialog(int id)      //创建对话框
    {
        Dialog result = null;
        switch(id)
        {
            case 0:
                 AlertDialog.Builder b = new AlertDialog.Builder(this);
                b.setItems(
                  null,
                  null
                );
                b.setCancelable(false);
                gmDialog = b.create();
                result = gmDialog;
            break;
        }
        return result;
    }
    @Override
    //每次弹出对话框时被回调,动态更新对话框内容的方法
    public void onPrepareDialog(int id,final Dialog dialog)    {
```

```
    switch(id)
    {
        case 0:
            dialog.setContentView(R.layout.dialog);
            Button bok = (Button)dialog.findViewById(R.id.bOk);
            Button bcancel = (Button)dialog.findViewById(R.id.bCancle);
            final EditText et = (EditText)dialog.findViewById(R.id.et);
            bok.setOnClickListener                              //确定按钮监听器
            (
                new OnClickListener()
                {
                    public void onClick(View arg0)
                    {
                String newName = et.getText().toString().trim(); //获取新文件名称
                        //获取修改后的文件路径
                        File xgf = new File(leavePath);
                        String newPath = xgf.getParentFile().getPath() + "/" + newName;
                        xgf.renameTo(new File(newPath));
                final File[] files = getFiles(currentPath);       //获取根节点文件列表
                    intoListView(files,lv);    //将各个文件添加到 ListView 列表中

                        dialog.cancel();
                    }
                }
            );
            bcancel.setOnClickListener          //取消按钮监听器
            (
                new OnClickListener()
                {
                    public void onClick(View arg0)
                    {
                        dialog.cancel();
                    }
                }
            );
            dialog.setCancelable(true);
        break;
    }
}
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
lv = (ListView)MainActivity.this.findViewById(R.id.lvwjlb);//获取 ListView 控件对象
    Button bgml = (Button)this.findViewById(R.id.bgml);     //搜索根目录文件按钮
    Button bsf = (Button)this.findViewById(R.id.bsf);       //搜索父目录文件按钮
    bgml.setOnClickListener                                 //搜索跟目录文件按钮监听器
    (
        new OnClickListener()
        {
            public void onClick(View v) {
```

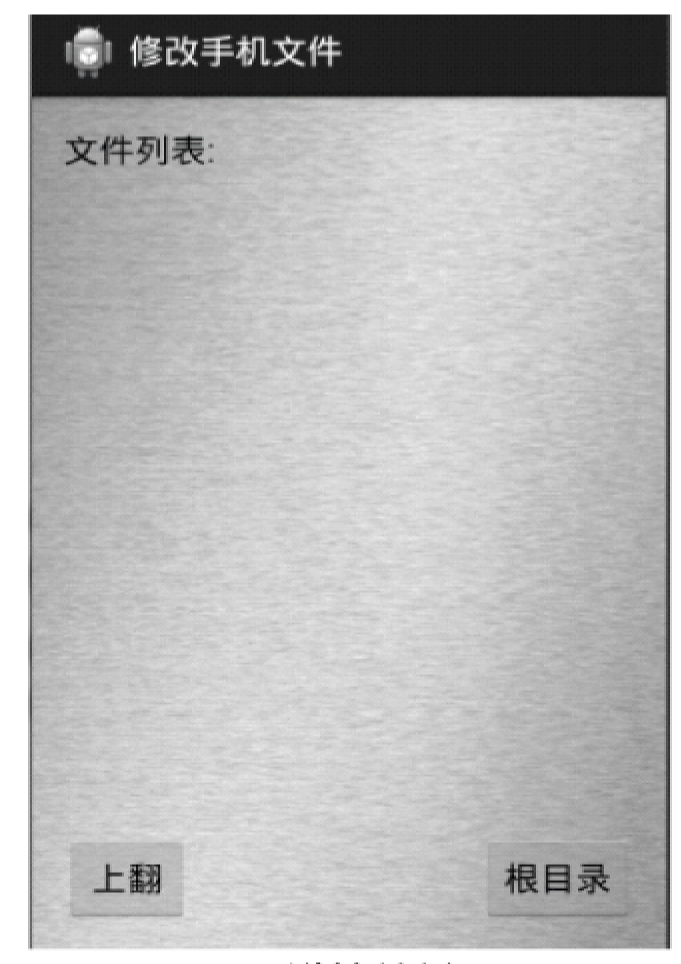
```
                    currentPath = rootPath;
                    final File[] files = getFiles(currentPath);  //获取根节点文件列表
                    intoListView(files,lv);             //将各个文件添加到 ListView 列表中
                }
            }
        );
        bsf.setOnClickListener                         //搜索父目录文件按钮监听器
        (
            new OnClickListener()
            {
                public void onClick(View v) {
                    if((currentPath!= null)&&(!currentPath.equals(rootPath)))
                {  //如果当前父路径不是 rootPath,则单击上翻键,回到上一层目录
                File cf = new File(currentPath);       //获取当前文件列表的路径对应的文件
                        cf = cf.getParentFile();       //获取父目录文件
                        currentPath = cf.getPath();  //记录当前文件列表路径

                        intoListView(getFiles(currentPath),lv);
                    }
                }
            }
        );
    }
    //获取当前目录下的文件列表
    public File[] getFiles(String filePath)
    {
        File[] files = new File(filePath).listFiles();//获取当前目录下的文件
        return files;
    }
    //将文件列表添加到 ListView 中
    public void intoListView(final File[] files,final ListView lv)
    {
        if(files!= null)                               //当文件列表不为空时
        {
            if(files.length == 0)
            {                                          //当前目录为空
                File cf = new File(currentPath);       //获取当前文件列表的路径对应的文件
                cf = cf.getParentFile();               //获取父目录文件
                currentPath = cf.getPath();            //记录当前文件列表路径
                Toast.makeText(MainActivity.this,"该文件夹为空!",Toast.LENGTH_SHORT).show();
            }
            else
            {
                BaseAdapter ba = new BaseAdapter()     //创建适配器
                {
                    public int getCount() {
                        return files.length;
                    }
                    public Object getItem(int position) {
                        return null;
                    }
```

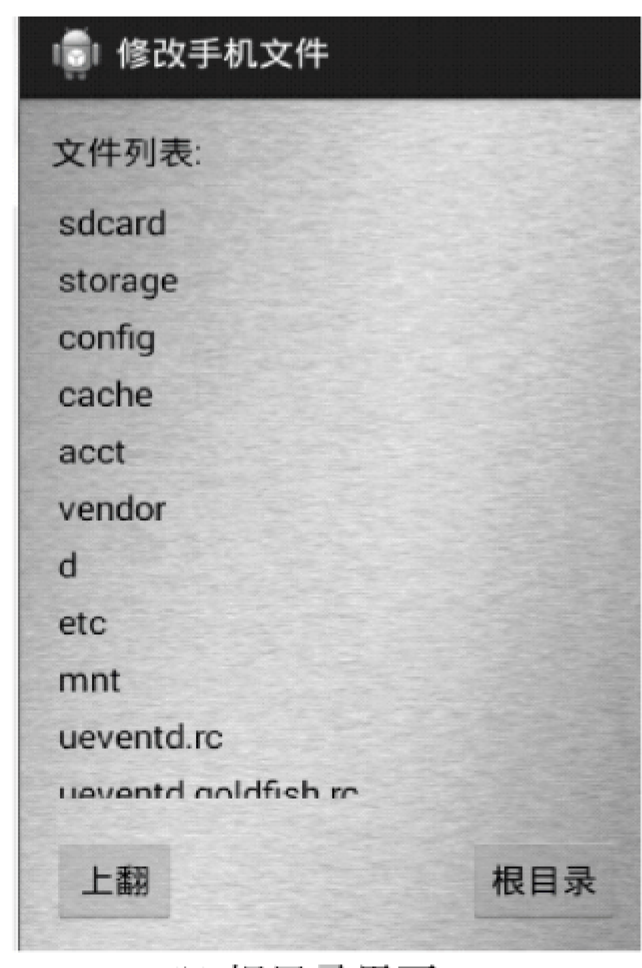
```
                public long getItemId(int position) {
                    return 0;
                }
                public View getView(int arg0,View arg1,ViewGroup arg2) {
                    LinearLayout ll = new LinearLayout(MainActivity.this);
                    ll.setOrientation(LinearLayout.VERTICAL);    //竖直排列
                    ll.setPadding(5,5,5,5);                      //留白
                TextView tv = new TextView(MainActivity.this);   //初始化 TextView
                    tv.setTextColor(Color.BLACK);                //设置字体颜色
                    tv.setText(files[arg0].getName());           //添加文件名称
                    tv.setGravity(Gravity.LEFT);                 //左对齐
                    tv.setTextSize(18);                          //字体大小
                    ll.addView(tv);                          //LinearLayout 添加 TextView
                    return ll;
                }
            };
            lv.setAdapter(ba);                               //设置适配器

            lv.setOnItemClickListener                        //设置选中菜单的监听器
            (
                new OnItemClickListener()
                {
                    public void onItemClick(AdapterView<?> arg0,View arg1,
                            int arg2,long arg3) {
            File f = new File(files[arg2].getPath());        //获得当前单击的文件对象
                        if(f.isDirectory())                  //存在分支
                        {
                            currentPath = files[arg2].getPath();
                File[] fs = getFiles(currentPath);           //获取当前路径下所有子文件
                        intoListView(fs,lv);                 //将子文件列表填入 ListView 中
                        }
                        else
                        {                              //弹出对话框,供用户填写新的文件名称
                            leavePath = f.getPath();
                            showDialog(0);
                        }
                    }
                }
            );
        }
    }
    else
    {
        File cf = new File(currentPath);             //获取当前文件列表的路径对应的文件
        cf = cf.getParentFile();                     //获取父目录文件
        currentPath = cf.getPath();                  //记录当前文件列表路径
        Toast.makeText(MainActivity.this,"该文件夹为空!",Toast.LENGTH_SHORT).show();
    }
  }
}
```

运行程序,效果如图 8-4(a)所示,当单击界面中的“根目录”按钮,效果如图 8-4(b)所示,单击图 8-4(b)中对应的文件,效果如图 8-4(c)所示。



(a) 默认效果



(b) 根目录界面



(c) 修改文件名称

图 8-4　修改手机文件

8.3　SQLite 存储数据

Android 系统提供了 SQLite 标准的数据库,完整支持的 SQL 语句。同样,它可以保存较大的数据,既可以保存在系统中也可以保存在 SDcard 中。数据库存储具有一定规范的数据是非常高效的,但是需要相应数据库的操作规范,相对前两个较复杂。

使用 SQLite 数据库的步骤为:

(1) 创建数据库;

(2) 打开数据库;

(3) 创建表;

(4) 完成数据的增、删、改、查操作;

(5) 关闭数据库。

下面通过一个实例来演示 SQLite 存储数据。

【例 8-5】 使用 SQLite 实现数据库的添加、删除、查找、修改等操作。其实现操作为:

(1) 在 Eclipse 环境下建立一个名为 SQLite_test 的工程。

(2) 编写布局文件,布局 4 个按钮、3 个文本框及两个编辑框控件。打开 res/Layout 目录下的 main.xml 文件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
android:orientation = "vertical"
android:layout_width = "fill_parent"
android:layout_height = "fill_parent"
android:background = "@drawable/h">
< Button
```

```
android:text = "添加"
android:id = "@ + id/main_btn_insert"
android:layout_width = "wrap_content"
android:layout_height = "wrap_content"/>
< Button
android:text = "删除"
android:id = "@ + id/main_btn_delete"
android:layout_width = "wrap_content"
android:layout_height = "wrap_content"/>
< Button
android:text = "查询"
android:id = "@ + id/main_btn_select"
android:layout_width = "wrap_content"
android:layout_height = "wrap_content"/>
< Button
android:text = "更新"
android:id = "@ + id/main_btn_update"
android:layout_width = "wrap_content"
android:layout_height = "wrap_content"/>
< TextView
android:text = "姓名"
android:layout_width = "wrap_content"
android:layout_height = "wrap_content"/>
< EditText
android:id = "@ + id/main_et_name"
android:layout_width = "fill_parent"
android:layout_height = "wrap_content"/>
< TextView
android:text = "性别"
android:layout_width = "wrap_content"
android:layout_height = "wrap_content"/>
< EditText
android:id = "@ + id/main_et_sex"
android:layout_width = "fill_parent"
android:layout_height = "wrap_content"/>
< TextView
android:id = "@ + id/main_tv_showContent"
android:layout_width = "wrap_content"
android:layout_height = "wrap_content"/>
</LinearLayout >
```

(3) 继承自 SQLiteOpenHelper，在 src/ com. example. sqlite_e 包下创建一个名称为 E_OpenHelper 的文件，用来打开或创建一个数据库。代码为：

```
package com. example. sqlite_test;
import android. content. Context;
import android. database. sqlite. SQLiteDatabase;
import android. database. sqlite. SQLiteDatabase. CursorFactory;
import android. database. sqlite. SQLiteOpenHelper;
public class E_OpenHelper extends SQLiteOpenHelper
{
```

```
    String sql = "create table if not exists TestUsers" + "(id int primary key,name varchar,sex
varchar)";
    public E_OpenHelper(Context context,String name,CursorFactory factory,int version)
    {
        super(context,name,factory,version);
        //TODO:自动存根法
    }
    @Override
    public void onCreate(SQLiteDatabase db)
    {
        //TODO:自动存根法
        db.execSQL(sql);
    }
    @Override
    public void onUpgrade(SQLiteDatabase arg0,int arg1,int arg2)
    {   //TODO:自动存根法
    }
}
```

(4) 编写 Activity 文件,用于实现数据库的添加、删除、查找、修改等操作。打开 src/com.example.sqlite_test 包下的 MainActivity.java 文件。代码为:

```
package com.example.sqlite_test;
import android.app.Activity;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.*;
public class MainActivity extends Activity
{
    Button btnInsert;
    Button btnDelete;
    Button btnUpdate;
    Button btnSelect;
    EditText etName;
    EditText etSex;
    TextView tvShowContent;
    E_OpenHelper OpenHelper;
    SQLiteDatabase db = null;
    public static final String DB_NAME = "DBTest";
    View.OnClickListener btnInsertListener = new View.OnClickListener()
    {
        @Override
        public void onClick(View v)
        {
            InsertTb();
        }
    };
```

```
View.OnClickListener btnDeleteListener = new View.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
       DeleteTb();
    }
};
View.OnClickListener btnUpdateListener = new View.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        UpdateTb();
    }
};
View.OnClickListener btnSelectListener = new View.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        Select();
    }
};
/** 第 1 次调用活动. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    OpenHelper = new E_OpenHelper(this,DB_NAME,null,1);
    btnInsert = (Button) findViewById(R.id.main_btn_insert);
    btnDelete = (Button) findViewById(R.id.main_btn_delete);
    btnUpdate = (Button) findViewById(R.id.main_btn_update);
    btnSelect = (Button) findViewById(R.id.main_btn_select);
    tvShowContent = (TextView) findViewById(R.id.main_tv_showContent);
    etName = (EditText) findViewById(R.id.main_et_name);
    etSex = (EditText) findViewById(R.id.main_et_sex);
    btnInsert.setOnClickListener(btnInsertListener);
    btnDelete.setOnClickListener(btnDeleteListener);
    btnUpdate.setOnClickListener(btnUpdateListener);
    btnSelect.setOnClickListener(btnSelectListener);
}
public void InsertTb()
{
    int flag = -1;
    db = OpenHelper.getWritableDatabase();
    String strName = etName.getText().toString();
    String strSex = etSex.getText().toString();
    String sql = "insert into TestUsers (name,sex) values ('" + strName + "','" + strSex + "')";
    try {
```

```
            db.execSQL(sql);
            } catch (SQLException e)
            {
                Log.i("err","insert failed");
                flag = 0;
                Toast.makeText(MainActivity.this,"插入失败!",Toast.LENGTH_SHORT).show();
            }
            db.close();
            if (flag == -1)
            {
                Toast.makeText(MainActivity.this,"插入成功!",Toast.LENGTH_SHORT).show();
            }
        }
    public void DeleteTb()
    {
        int flag = -1;
        db = OpenHelper.getWritableDatabase();
        String sql = "delete from TestUsers where id = 2";
        try {
            db.execSQL(sql);
            } catch (SQLException e)
            {
                Log.i("err","delete failed");
                flag = 0;
                Toast.makeText(MainActivity.this,"删除失败!",Toast.LENGTH_SHORT).show();
            }
        db.close();
        if (flag == -1)
        {
            Toast.makeText(MainActivity.this,"删除成功!",Toast.LENGTH_SHORT).show();
        }
    }
    public void UpdateTb()
    {
        int flag = -1;
        db = OpenHelper.getWritableDatabase();
        String Name = etName.getText().toString();
        String sql = "Update TestUsers set name = 'anhong',sex = 'men' where name = '" + Name + "'";
        try {
            db.execSQL(sql);
            } catch (SQLException e)
            {
                Log.i("err","update failed");
                flag = 0;
                Toast.makeText(MainActivity.this,"更新失败!",Toast.LENGTH_SHORT).show();
            }
            db.close();
            if (flag == -1)
            {
                Toast.makeText(MainActivity.this,"更新成功!",Toast.LENGTH_SHORT).show();
            }
```

```
        }
        public void Select()
        {
            db = OpenHelper.getReadableDatabase();
            String sql = "select sex from TestUsers where name = ?";
            Cursor cursor = db.rawQuery(sql,new String[]
            { etName.getText().toString()
            });
            int count = cursor.getCount();
            String [] Sex = new String[count];
            int i = 0;
            if (cursor.getCount() > 0)
            {
                //取多条记录
                int sexIndex = cursor.getColumnIndex("sex");
                for(cursor.moveToFirst();!(cursor.isAfterLast());cursor.moveToNext())
                {
                    Sex[i] = cursor.getString(sexIndex);
                    i++;
                }
            }
            for(int j = 0; j < count; j++)
            {
                tvShowContent.append("");
                tvShowContent.append(Sex[j]);
            }
        }
    }
```

运行程序,效果如图 8-5 所示。



图 8-5　SQLite 数据存储操作

8.4 ContentProvider 数据共享

ContentProvider 内部怎样保存数据由其设计者决定。但是所有的 ContentProvider 都实现一组通用的方法,用来提供数据的增、删、改、查功能。

客户端通常不会直接使用这些应运,大多数是通过 ContentResolver 对象实现对 ContentProvider 的操作。开发人员可以通过调用 Activity 或者其他应用程序组件的实现类中的 getContentRsolver()方法获得 ContentProvider 对象,例如:

```
ContentResolver cr = getContentResolver();
```

使用 ContentResolver 提供的方法可以获得 ContentProvider 中任何感兴趣的数据。

当开始查询时,Android 系统确认查询的目标 ContentProvider 并确保它正在运行。系统会初始化所有 ContentProvider 类的对象,开发人员不必完成此类操作。实际上,开发人员根本不会直接使用 ContentProvider 类的对象。通常,每个类型的 ContentProvider 仅有一个单独的实例。但是该实例能与位于不同应用程序和进程的多个 ContentResolver 类对象通信。不同进程之间的通信由 ContentProvider 类和 ContentResolver 类处理。

Android 系统提供的常见 ContentProvider 说明如下。

- Browser:读取或修改书签、浏览历史或网络搜索。
- CallLog:查看或更新通话历史。
- Contacts:获取、修改或保存联系人信息。
- LiveFolders:由 ContentProvider 提供内容的特定文件夹。
- MediaStore:访问声明、视频和图片。
- Setting:查看和获取蓝牙设置、铃声和其他设备偏好。
- SearchRecentSuggestions:能被配置以使用查找意见的 provider 操作。
- SynStateContract:用于使用数据数组账号关联数据的 ContentProvider 约束。希望使用标准方式保存数据的 provider 可以使用它。
- UserDictionary:在可预测文本输入时,提供用户定义单词给输入法使用。应用程序和输入法能增加数据到该字典。单词能关联频率信息和本地化信息。

8.4.1 数据模型

ContentProvider 使用基于数据库模型的简单表格来提供其中的数据,这里每行代表一条记录,每列代表特定类型和含义的数据。例如,联系人的信息可能以表 8-1 所示的方式提供。

表 8-1 联系方式

_ID	NAME	NUMBER	EMAIL
001	李 * *	127 * * * * *	a12 * * @163.com
002	刘 * *	276 * * * * *	b12 * * @126.com
003	陆 * *	311 * * * * *	c32 * * @qq.com
004	周 * *	324 * * * * *	d41 * * @google.com

每条记录包含一个数值型的_ID字段，它用于在表格中唯一标记该记录。ID能用于匹配相关表格中的记录，例如在一个表格中查询联系人电话，在另一表格中查询其工作经历。

注意：ID字段前还包含一个下划线，在编写代码时不要忘记。

查询返回一个Cursor对象，它能遍历各行各列来读取各个字段的值。对于各个类型的数据，它都提供了专用的方法。因此，为了读取字段的数据，开发人员必须知道当前字段包含的数据类型。

8.4.2 URI用法

实现URI的语法格式为：

```
Uri uri = Uri.parse("content://com.changcheng.provider.contactprovider/contact")
```

在Content Provider中使用的查询字符串有别于标准的SQL查询。很多例如select、add、delete、modify等操作都使用一种特殊的URI来进行，这种URI由3个部分组成，"content://"，代表数据的路径，和一个可选的标识数据的ID。以下是一些示例URI：

content://media/internal/images：这个URI将返回设备上存储的所有图片。

content://contacts/people/：这个URI将返回设备上的所有联系人信息。

content://contacts/people/45：这个URI返回单个结果(联系人信息中ID为45的联系人记录)。

尽管这种查询字符串格式很常见，但是它看起来还是有点令人疑惑。为此，Android提供一系列的帮助类(在android.provider包下)，里面包含了很多以类变量形式给出的查询字符串，这种方式更容易让人理解一点，因此，如上面content://contacts/people/45这个URI就可以写成如下形式：

```
Uri person = ContentUris.withAppendedId(People.CONTENT_URI,  45);
```

【例8-6】 本实例用于演示URI的用法。其具体实现步骤为：

(1) 在Eclipse中创建一个Android应用项目，命名为ContentURI_test。

(2) 打开res\layout目录下的main.xml文件，在文件中声明两个Button控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aabbcc">
    < Button
        android:text = "插入"
        android:id = "@ + id/button1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"/>
    < Button
        android:text = "删除"
        android:id = "@ + id/button2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"/>
```

```
</LinearLayout>
```

(3) 在 res\layout 目录下创建一个 insert. xml 布局文件,实现插入页面布局,在文件中声明一个 EditText 控件。代码为:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#bbccdd">
    <EditText
        android:text="insert"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/editText1"></EditText>
</LinearLayout>
```

(4) 在 res\layout 目录下创建一个 delete. xml 删除界面布局文件,在文件中声明一个 EditText 控件。代码为:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#cceeff">
    <EditText
        android:text="delete"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/editText1"/>
</LinearLayout>
```

(5) 打开 src\fs. contenturi_test 包下的 MainActivity. java 文件,在该文件中实现跳转到"插入"及"删除"界面。代码为:

```
package fs.contenturi_test;
import android.content.ContentProvider;
import android.content.ContentValues;
import android.content.UriMatcher;
import android.database.Cursor;
import android.net.Uri;
import android.util.Log;
public class MainActivity extends ContentProvider {
    private static final UriMatcher uriMatcherRef = new UriMatcher(
            UriMatcher.NO_MATCH);
    static {
        uriMatcherRef.addURI("com.gaohongyan.www","insert",1000);
        uriMatcherRef.addURI("com.gaohongyan.www","delete/#",2000);
    }
```

```
    @Override
    public String getType(Uri arg0) {
        Log.v("!","调用了 getType()方法");
        switch (uriMatcherRef.match(arg0)) {
        case 1000:
            Log.v("!","匹配了 insert");
            return "vnd.android.cursor.item/insert";
        case 2000:
            Log.v("!","匹配了 delete");
            return "vnd.android.cursor.item/delete";
        default:
            throw new IllegalArgumentException();
        }
    }
    @Override
    public int delete(Uri arg0,String arg1,String[] arg2) {
        //TODO 自动存根法
        return 0;
    }
    @Override
    public Uri insert(Uri arg0,ContentValues arg1) {
        //TODO 自动存根法
        return null;
    }
    @Override
    public boolean onCreate() {
        //TODO 自动存根法
        return false;
    }
    @Override
    public Cursor query(Uri arg0,String[] arg1,String arg2,String[] arg3,
            String arg4) {
        //TODO 自动存根法
        return null;
    }
    @Override
    public int update(Uri arg0,ContentValues arg1,String arg2,String[] arg3) {
        //TODO 自动存根法
        return 0;
    }
}
```

(6) 在 src\fs.contenturi_test 包下创建一个 Main.java 文件,该文件用于实现主界面。代码为:

```
package fs.contenturi_test;
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
```

```
import android.view.View.OnClickListener;
import android.widget.Button;
public class Main extends Activity {
    private Button button1;
    private Button button2;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        button1 = (Button) this.findViewById(R.id.button1);
        button1.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View arg0) {
                Intent intent = new Intent("insertAction",Uri
                        .parse("content://fs.contenturi_test.www/insert"));
                Main.this.startActivity(intent);
            }
        });
        button2 = (Button) this.findViewById(R.id.button2);
        button2.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View arg0) {
                Intent intent = new Intent("deleteAction",Uri
                        .parse("content://fs.contenturi_test.www/delete/999"));
                Main.this.startActivity(intent);
            }
        });
    }
}
```

(7) 在 src\fs.contenturi_test 包下创建一个 Insert.java 文件，该文件用于实现“插入界面”。代码为：

```
package fs.contenturi_test;
import android.app.Activity;
import android.os.Bundle;
public class Insert extends Activity {
    /** 第 1 次调用 activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.insert);
    }
}
```

(8) 在 src\fs.contenturi_test 包下创建一个 Delete.java 文件，该文件用于实现“删除界面”。代码为：

```
package fs.contenturi_test;
import android.app.Activity;
import android.os.Bundle;
```

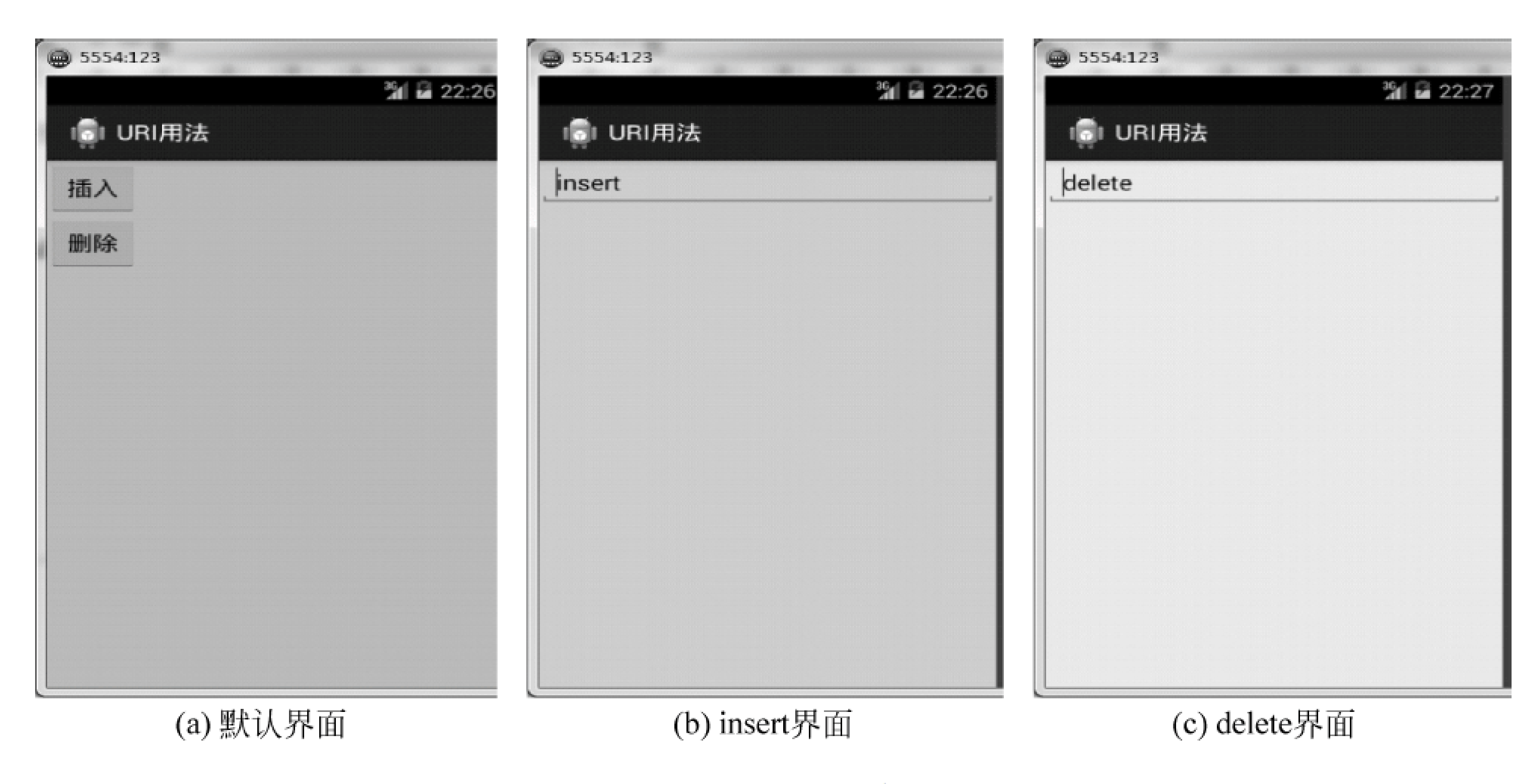
```
public class Delete extends Activity {
    /**  第 1 次调用 activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.delete);
    }
}
```

(9) 设置权限，打开 AndroidManifest.xml。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.contenturi_test"
    android:versionCode = "1"
    android:versionName = "1.0" >
    <uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
    <application
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        <activity
            android:name = ".Main"
            android:label = "@string/app_name" >
            <intent - filter>
                <action android:name = "android.intent.action.MAIN" />
                <category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter>
    </activity>
        <activity android:name = ".Delete" android:label = "@string/app_name">
            <intent - filter>
                <action android:name = "deleteAction"></action>
                <category android:name = "android.intent.category.DEFAULT" />
                <data android:mimeType = "vnd.android.cursor.item/delete"></data>
            </intent - filter>
        <! -- 设置权限 -->
        </activity>
        <activity android:name = ".Insert" android:label = "@string/app_name">
            <intent - filter>
                <action android:name = "insertAction"></action>
                <category android:name = "android.intent.category.DEFAULT" />
                <data android:mimeType = "vnd.android.cursor.item/insert"></data>
            </intent - filter>
        </activity>
        <provider android:name = ".MainActivity"
            android:authorities = "fs.contenturi_test.www"></provider>
    </application>
</manifest>
```

注意：fs. contenturi_test. www 主机名字符串必须与 AndroidMainfiest. xml 文件中的 <provider>节点 android:authorities 属性值一致，android:authorities 属性值的作用是标识一个 ContentProvider 对象在系统中的唯一。

运行程序，默认效果如图 8-6(a)所示，单击界面中的"插入"按钮即进入"Insert"界面，效果如图 8-6(b)所示，单击界面中的"删除"按钮即进入"delete"界面，效果如图 8-6(c)所示。



(a) 默认界面　　(b) insert界面　　(c) delete界面

图 8-6　URI 方法使用

8.4.3　ContentProvider 详析

一个程序可以通过实现一个 ContentProvider 的抽象接口将自己的数据完全暴露出去，而且 ContentProvider 是以类似数据库中表的方式将数据暴露，也就是说 ContentProvider 就像一个"数据库"。那么外界获取其提供的数据，也就应该与从数据库中获取数据的操作基本一样，只不过是采用 URI 来表示外界需要访问的"数据库"。

Content Provider 提供了一种多应用间数据共享的方式，例如：联系人信息可以被多个应用程序访问。

Content Provider 是个实现了一组用于提供其他应用程序存取数据的标准方法的类。应用程序可以在 Content Provider 中执行如下操作：查询数据、修改数据、添加数据、删除数据标准的 Content Provider。Android 提供了一些已经在系统中实现的标准 Content Provider，例如联系人信息、图片库等，你可以用这些 Content Provider 来访问设备上存储的联系人信息、图片等。

ContentProvider 提供的方法，用于实现其对应的基本操作如下：

- query：查询；
- insert：插入；
- update：更新；
- delete：删除；
- getType：得到数据类型；
- onCreate：创建数据时调用的回调函数。

1. 查询操作

ContentProvider 类通过以下语法格式实现查询操作：

```
Cursor cur = managedQuery(person,null,null,null);
```

这个查询返回一个包含所有数据字段的游标，可以通过迭代这个游标来获取所有的数据：

```
package com.wissen.testApp;
public class ContentProviderDemo extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        displayRecords();
    }
    private void displayRecords()
    {
        //该数组中包含了所有要返回的字段
        String columns[] = new String[] { People.NAME,People.NUMBER };
        Uri mContacts = People.CONTENT_URI;
        Cursor cur = managedQuery(
            mContacts,
            columns,  //要返回的数据字段
            null,                         //WHERE 子句
            null,                         //WHERE 子句的参数
            null                          //Order - by 子句
      );
    if (cur.moveToFirst())
    {
            String name = null;
            String phoneNo = null;
            do {
                //获取字段的值
                name = cur.getString(cur.getColumnIndex(People.NAME));
                phoneNo = cur.getString(cur.getColumnIndex(People.NUMBER));
                Toast.makeText(this,name + " " + phoneNo,Toast.LENGTH_LONG).show();
            } while (cur.moveToNext());
        }
    }
}
```

以上代码演示了一个怎样依次读取联系人信息表中的指定数据列 name 和 number。

注意：实现 Content Provider 查询有两个方法，分别为 ContentResolver 的 query()和 Activity 对象的 managedQuery()。二者接收的参数均相同，返回的都是 Cursor 对象，唯一不同的是，使用 managedQuery 方法可以让 Activity 来管理 Cursor 的生命周期。被管理的 Cursor 会在 Activity 进入暂停状态的时候调用自己的 deactivate 方法自行卸载，而在

Activity 回到运行状态时会调用自己的 requery 方法重新查询生成的 Cursor 对象。如果一个未被管理的 Cursor 对象想被 Activity 管理，可以调用 Activity 的 startManagingCursor 方法来实现。

2. 插入操作

在 Android 的 Content Provider 中，可以使用 ContentResolver. update()方法来修改数据，以下代码为编写一个修改数据的方法：

```
 private void updateRecord(int recNo, String name)
{
    Uri uri = ContentUris.withAppendedId(People.CONTENT_URI, recNo);
    ContentValues values = new ContentValues();
    values.put(People.NAME, name);
    getContentResolver().update(uri, values, null, null);
}
```

现在可以调用上面的方法来更新指定记录：

```
updateRecord(10, "ABC");            //更改第 10 条记录的 name 字段值为"ABC"
```

3. 添加操作

要增加记录，可以调用 ContentResolver. insert()方法，该方法接收一个要增加的记录的目标 URI 以及一个包含了新记录值的 Map 对象，调用后的返回值是新记录的 URI，包含记录号。

以下代码用于创建一个 insertRecord()方法以对联系人信息簿中进行数据的添加：

```
private void insertRecords(String name, String phoneNo)
{
    ContentValues values = new ContentValues();
    values.put(People.NAME, name);
    Uri uri = getContentResolver().insert(People.CONTENT_URI, values);
    Log.d("ANDROID", uri.toString());
    Uri numberUri = Uri.withAppendedPath(uri, People.Phones.CONTENT_DIRECTORY);
    values.clear();
    values.put(Contacts.Phones.TYPE, People.Phones.TYPE_MOBILE);
    values.put(People.NUMBER, phoneNo);
    getContentResolver().insert(numberUri, values);
}
```

这样就可以调用 insertRecords(name，phoneNo)的方式来向联系人信息簿中添加联系人姓名和电话号码了。

4. 删除操作

ContentProvider 中的 getContextResolver. delete()方法可以用来删除记录，下面的记录用来删除设备上所有的联系人信息：

```
private void deleteRecords()
{
    Uri uri = People.CONTENT_URI;
```

```
    getContentResolver().delete(uri,null,null);
}
```

也可以指定 WHERE 条件语句来删除特定的记录：

```
getContentResolver().delete(uri,"NAME = " + "'XYZ XYZ'",null);
```

这将会删除 name 为'XYZ XYZ'的记录。

下面通过一个实例来演示 ContentProvider 的基本操作。

【例 8-7】 实现自动补全联系人姓名的功能。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Contact_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中设置背景图片、标签属性并增加一个自动补全标签。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/bj2"
    android:orientation = "vertical" >
    < TextView
        android:id = "@ + id/title"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_gravity = "center"
        android:text = "自动补全联系人姓名"
        android:textColor = "@android:color/black"
        android:textSize = "30dp" />
    < LinearLayout
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:orientation = "horizontal" >

        < TextView
            android:id = "@ + id/textView"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_margin = "5dp"
            android:text = "姓名: "
            android:textColor = "@android:color/black"
            android:textSize = "25dp" />
        < AutoCompleteTextView
            android:id = "@ + id/edit"
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
            android:completionThreshold = "1"
            android:textColor = "@android:color/black" >
            < requestFocus />
        </AutoCompleteTextView >
    </LinearLayout >
```

```
</LinearLayout>
```

(3) 打开 src\fs.contact_test 包下的 MainActivity.java 文件，该类继承了 Activity 类，在重写 onCreate()方法时，完成自动补全的设置。代码为：

```
package fs.contact_test;
import android.app.Activity;
import android.content.ContentResolver;
import android.database.Cursor;
import android.os.Bundle;
import android.provider.ContactsContract.Contacts;
import android.widget.AutoCompleteTextView;
public class MainActivity extends Activity {
    private  String[] columns = new String[] {
Contacts._ID,Contacts.DISPLAY_NAME };
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ContentResolver resolver = getContentResolver();
      Cursor cursor = resolver.query(Contacts.CONTENT_URI,columns,null,null,null);
        ContactAdapter adapter = new ContactAdapter(this,cursor);
        AutoCompleteTextView textView = (AutoCompleteTextView) findViewById(R.id.edit);
        textView.setAdapter(adapter);
    }
}
```

(4) 在 src\fs.contact_test 包下创建一个 ContactAdapter 类，它继承了 CursorAdapter 类并实现了 Filterable 接口，在重写方法时完成了获取联系人姓名的功能。代码为：

```
package fs.contact_test;
import android.content.ContentResolver;
import android.content.Context;
import android.database.Cursor;
import android.net.Uri;
import android.provider.ContactsContract.Contacts;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CursorAdapter;
import android.widget.FilterQueryProvider;
import android.widget.Filterable;
import android.widget.TextView;
public class ContactAdapter extends CursorAdapter implements Filterable {
    private ContentResolver resolver;
    private  String[] columns = new String[] { Contacts._ID,Contacts.DISPLAY_NAME };
    public ContactAdapter(Context context,Cursor c) {
        super(context,c);                           //调用父类构造方法
        resolver = context.getContentResolver();    //初始化 ContentResolver
    }
    @Override
```

```
    public void bindView(View arg0,Context arg1,Cursor arg2) {
        ((TextView) arg0).setText(arg2.getString(1));
    }
    @Override
    public View newView(Context context,Cursor cursor,ViewGroup parent) {
        LayoutInflater inflater = LayoutInflater.from(context);
        TextView view = (TextView) inflater.inflate(android.R.layout.simple_dropdown_item_
1line,parent,false);
        view.setText(cursor.getString(1));
        return view;
    }
    @Override
    public CharSequence convertToString(Cursor cursor) {
        return cursor.getString(1);
    }
    @Override
    public Cursor runQueryOnBackgroundThread(CharSequence constraint) {
        FilterQueryProvider filter = getFilterQueryProvider();
        if (filter != null) {
            return filter.runQuery(constraint);
        }
        Uri uri = Uri.withAppendedPath(Contacts.CONTENT_FILTER_URI,Uri.encode(constraint.
toString()));
        return resolver.query(uri,columns,null,null,null);
    }
}
```

(5) 打开 AndroidManifest.xml 文件，在文件中增加读取联系人记录权限。代码为：

```
<?xml version = "1.0" encoding = "utf-8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.contact_test"
    android:versionCode = "1"
    android:versionName = "1.0" >
    <uses-sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
   <!-- 添加权限 -->
    <uses-permission android:name = "android.permission.READ_CONTACTS"/>
    <application
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        <activity
            android:name = "fs.contact_test.MainActivity"
            android:label = "@string/app_name" >
            <intent-filter>
                <action android:name = "android.intent.action.MAIN" />
                <category android:name = "android.intent.category.LAUNCHER" />
            </intent-filter>
```

```
        </activity>
    </application>
</manifest>
```


运行程序,效果如图8-7所示。



图8-7　自动补全联系人姓名

第 9 章　Android 经典应用

在前面已经对 Android 的布局、控件、绘图、动画、菜单、对话框及数据存储共享等基本内容进行了介绍，本节将概括地介绍 Android 的实际应用。

9.1　Android 多媒体技术

Android 中的多媒体是基于第三方 PacketVideo 公司的 OpenCore 来实现的，支持所有通用的音频、视频、静态图像格式，包括 MPEG4、H. 264、MP3、AAC、AMR、JPG、PNG、GIF 等。从功能上分为两部分，一是音/视频的回放(PlayBack)，二是音视频的纪录(Recorder)。OpenCore 是 Android 多媒体的核心，OpenCore 多媒体框架有一套通用可扩展的接口针对第三方的多媒体编码器、输入、输出设备等，支持多媒体文件的播放、下载。

9.1.1　Android 音频

在多媒体播放中，Android 系统使用了一个名为 MediaPlayer 的类。该类可以用来播放音频、视频和流媒体，MediaPlayer 包含了音频(Audio)和视频(Video)的播放功能。

Android 系统支持 3 种不同来源的音频播放：

(1) 本地资源

存储在应用程序中的资源，例如存储在 RAW 文件夹下的媒体文件，只能被当前应用程序访问。

(2) 外部资源

存储在文件系统中的标准媒体文件，例如存储在 SD 卡中的文件，可以被所有应用程序访问。

(3) 网络资源

通过网络地址取得的数据流(URL)，例如"http://www. musiconline. com/classic/05. mp3"，可以被所有应用程序访问。

下面通过一个实例来演示音频播放器。

【例 9-1】 演示一个音乐播放实例。

在这里面除了调用 MediaPlayer 的 API 外，还需要处理当播放音乐时遇到来电等事件时的情况，要保证接听完电话后还能继续播放音乐，需要覆写 Activity 的生命周期的几个方法。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 MediaPlayerAudio_test。

(2) 打开 res\layout 目录下的 main. xml 布局文件，在文件中添加几个按钮控件及一个

编辑框控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/bj">
<TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "音乐文件" />
    <EditText
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "kaka.mp3"
        android:id = "@ + id/filename"/>
    <LinearLayout
        android:orientation = "horizontal"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content">
        <Button
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "播放"
            android:id = "@ + id/play"/>
        <Button
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "暂停"
            android:id = "@ + id/pause"/>
        <Button
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "重播"
            android:id = "@ + id/reset"/>
        <Button
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "停止"
            android:id = "@ + id/stop" />
    </LinearLayout>
</LinearLayout>
```

(3) 打开 src\fs.mediaplayer_test 包下的 MainActivity.java 文件，在文件中实现音乐的播放。代码为：

```
package fs.mediaplayeraudio_test;
import java.io.File;
import java.io.IOException;
import android.app.Activity;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.os.Environment;
```

```java
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
public class MainActivity extends Activity {
    private static final String TAG = "AudioPlayerActivity";
    private EditText filenameText;
    private MediaPlayer mediaPlayer;
    private String filename;
    private int position;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        filenameText = (EditText)this.findViewById(R.id.filename);
        mediaPlayer = new MediaPlayer();
        ButtonClickListener listener = new ButtonClickListener();
        Button playButton = (Button)this.findViewById(R.id.play);
        Button pauseButton = (Button)this.findViewById(R.id.pause);
        Button resetButton = (Button)this.findViewById(R.id.reset);
        Button stopButton = (Button) this.findViewById(R.id.stop);
        playButton.setOnClickListener(listener);
        pauseButton.setOnClickListener(listener);
        resetButton.setOnClickListener(listener);
        stopButton.setOnClickListener(listener);
        Log.i(TAG,"onCreate()");
    }
    @Override
    protected void onRestoreInstanceState(Bundle savedInstanceState) {
        this.filename = savedInstanceState.getString("filename");
        this.position = savedInstanceState.getInt("position");
        super.onRestoreInstanceState(savedInstanceState);
        Log.i(TAG,"onRestoreInstanceState()");
    }
    @Override
    protected void onSaveInstanceState(Bundle outState) {
        outState.putString("filename",filename);
        outState.putInt("position",position);
        super.onSaveInstanceState(outState);
        Log.i(TAG,"onSaveInstanceState()");
    }
    @Override
    protected void onPause() {                          //如果突然电话到来,则停止播放音乐
        if(mediaPlayer.isPlaying()){
            position = mediaPlayer.getCurrentPosition();  //保存当前播放点
            mediaPlayer.stop();
        }
        super.onPause();
    }
    @Override
    protected void onResume() {
        if(position > 0 && filename!= null){                //如果电话结束,则继续播放音乐
            try {
                play();
```

```
                mediaPlayer.seekTo(position);
                position = 0;
            } catch (IOException e) {
                Log.e(TAG,e.toString());
            }
        }
        super.onResume();
    }
    @Override
    protected void onDestroy() {
        mediaPlayer.release();
        super.onDestroy();
        Log.i(TAG,"onDestroy()");
    }
    private final class ButtonClickListener implements View.OnClickListener{
        @Override
        public void onClick(View v) {
            filename = filenameText.getText().toString(); //先得到文本框中的内容
            Button button = (Button) v;                   //得到 Button
            try {
                switch (v.getId()) {          //通过传过来的 Buttonid 可以判断 Button 的类型
                case R.id.play:               //播放
                    play();
                    break;
                case R.id.pause:
                    if(mediaPlayer.isPlaying()){
                      //让这个按钮上的文字显示为继续
                        mediaPlayer.pause();
                        button.setText(R.string.continue1);
                    }else{
                        mediaPlayer.start();//继续播放
                        button.setText(R.string.pause);
                    }
                    break;
                case R.id.reset:
                    if(mediaPlayer.isPlaying()){
                        mediaPlayer.seekTo(0);//让它从 0 开始播放
                    }else{
                        play();                //如果它没有播放,则让它开始播放
                    }
                    break;
                case R.id.stop:
                  //如果它正在播放,则让他停止
                    if(mediaPlayer.isPlaying()) mediaPlayer.stop();
                    break;
                }
            } catch (Exception e) {            //抛出异常
                Log.e(TAG,e.toString());
            }
        }
    }
    private void play() throws IOException {
        File audioFile = new File(Environment.getExternalStorageDirectory(),filename);
        mediaPlayer.reset();
```

```
            mediaPlayer.setDataSource(audioFile.getAbsolutePath());
            mediaPlayer.prepare();
            mediaPlayer.start();                    //播放
        }
    }
```

运行程序，效果如图 9-1 所示。



图 9-1 音乐播放

在 Android 开发中经常使用 MediaPlayer 来播放音频文件，但是 MediaPlayer 存在一些不足，例如：资源占用量较高、延迟时间较长、不支持多个音频同时播放等。这些缺点决定了 MediaPlayer 在某些场合的使用情况不会很理想，例如，在对时间精准度要求相对较高的游戏开发中。

在游戏开发中经常需要播放一些游戏音效(例如：子弹爆炸、物体撞击等)，这些音效的共同特点是短促、密集、延迟程度小。在这样的场景下，可以使用 SoundPool 代替 MediaPlayer 来播放这些音效。

SoundPool(android.media.SoundPool)，主要用于播放一些较短的声音片段，支持从程序的资源或文件系统加载。与 MediaPlayer 相比，SoundPool 的优势在于 CPU 资源占用量低和反应延迟小。另外，SoundPool 还支持自行设置声音的品质、音量、播放比率等参数，支持通过 ID 对多个音频流进行管理。

调用 SoundPool 对象的 play()方法可以播放指定音频。play()方法的格式为：

```
play(int soundID, float leftVolume, float rightVolume, int priority, int loop, float rate)
```

其中，参数 soundID 用于指定要播放的音频，该音频为通过 load()方法返回的音频；参数 leftVolume 用于指定左声道的音量，取值范围为 0.0～1.0；参数 rightVolume 用于指定

右声道的音量，取值范围为 0.0～1.0；参数 priority 用于指定播放音频的优先级，数值越大，优先级越高；参数 loop 用于指定播放循环次数，0 为不循环，－1 为循环；参数 rate 用于指定速率，1 为正常，最低为 0.5，最高为 2。

下面通过一个实例来演示 SoundPool 的用法。

【例 9-2】 利用 SoundPool 播放音频。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 SoundPool_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中添加 4 个 Button 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:background = "#ccddee"
    android:orientation = "vertical" >
    < Button
        android:id = "@ + id/button1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "风铃声" />
    < Button
        android:id = "@ + id/button2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_gravity = "clip_horizontal"
        android:text = "布谷鸟叫声" />
  < Button
        android:id = "@ + id/button3"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "门铃声" />
    < Button
        android:id = "@ + id/button4"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "电话声" />
</LinearLayout >
```

(3) 打开 src\fs.soundpool_test 包下的 MainActivity.java 文件，在文件中实现利用 soundPool 类实现音频播放。代码为：

```
package fs.soundpool_test;
import java.util.HashMap;
import android.app.Activity;
import android.media.AudioManager;
import android.media.SoundPool;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
```

```
import android.view.View.OnClickListener;
import android.widget.Button;
public class MainActivity extends Activity {
    private SoundPool soundpool;                                //声明一个 SoundPool 对象
    private HashMap<Integer,Integer> soundmap = new HashMap<Integer,Integer>();
    //创建一个 HashMap 对象
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button chimes = (Button) findViewById(R.id.button1);    //获取"风铃声"按钮
    Button enter = (Button) findViewById(R.id.button2);         //获取"布谷鸟叫声"按钮
        Button notify = (Button) findViewById(R.id.button3);    //获取"门铃声"按钮
        Button ringout = (Button) findViewById(R.id.button4);   //获取"电话声"按钮
        //创建一个 SoundPool 对象,该对象可以容纳 5 个音频流
     soundpool = new SoundPool(5,AudioManager.STREAM_SYSTEM,0);
        //将要播放的音频流保存到 HashMap 对象中
        soundmap.put(1,soundpool.load(this,R.raw.chimes,1));
        soundmap.put(2,soundpool.load(this,R.raw.enter,1));
        soundmap.put(3,soundpool.load(this,R.raw.notify,1));
        soundmap.put(4,soundpool.load(this,R.raw.ringout,1));
        soundmap.put(5,soundpool.load(this,R.raw.ding,1));
        //为各按钮添加单击事件监听器
        chimes.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                soundpool.play(soundmap.get(1),1,1,0,0,1);      //播放指定的音频
            }
        });
        enter.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                soundpool.play(soundmap.get(2),1,1,0,0,1);      //播放指定的音频
        }
        });
        notify.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                soundpool.play(soundmap.get(3),1,1,0,0,1);      //播放指定的音频
            }
        });
        ringout.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                soundpool.play(soundmap.get(4),1,1,0,0,1);      //播放指定的音频
        soundpool.play(soundpool.load(MainActivity.this,R.raw.notify,1),1,1,0,0,1);
            }
        });
    }
    //重写键被按下的事件
    @Override
```

```
        public boolean onKeyDown(int keyCode,KeyEvent event) {
            soundpool.play(soundmap.get(5),1,1,0,0,1);           //播放按键音
            return true;
        }
    }
```

(4) 在 res 中创建一个 raw 文件夹,在文件中放置 4 种类型的音频文件。

运行程序,效果如图 9-2 所示,当单击界面中的按钮时,即播放相应的声音。



图 9-2 SoundPool 播放音频

9.1.2 Android 后台播放音频

一般使用音频播放功能,只需要在 Activity 中新建一个 MediaPlayer 对象,便可以实现音频播放功能。但是,这并不能实现后台播放功能,因为一旦按了"Home"键,或者退出。该 MediaPlayer 播放音频完毕后,便不会再继续播放。因为播放完毕后,MediaPlayer 不能自动播放下一首音频。于是,使用了后台服务。

现在比较流行的作法是在 Service 中新建一个 MediaPlayer 对象,因为服务没有退出的话,服务的代码便可以继续执行。在服务中的 MediaPlayer 设置了 OnCompleteListenner,一旦播放完毕,便可以继续播放下一首。

下面通过一个实例来演示在 Android 中实现后台播放音频。

【例 9-3】 本实例用于实现当单击界面中的"Service"按钮时,即当前 Activity 结束,应用程序界面消失,返回 Android 应用程序列表,同时后台启动 Service,播放音频文件。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 AudioService_test。

(2) 打开 res\layout 目录下的 main. xml 布局文件，在文件中添加一个 Button 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = " # aabbcc">
    < Button
        android:id = "@ + id/button1"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:layout_gravity = "center_vertical"
        android:text = "启动 Service" />
 </LinearLayout >
```

(3) 打开 src\fs. audioservice_test 包下的 MainActivity. java 文件，在文件中实现启动 Activity 活动。代码为：

```
package fs.audioservice_test;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
public class MainActivity extends Activity {
    //第 1 次调用 Activity 活动
    private Button bt;
     @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        bt = (Button)findViewById(R.id.button1);
        bt.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //TODO 自动生成的方法存根
        startService(new Intent ("fs.audioservice_test.MY_AUDIO_SERVICE"));
                finish();
            }
        });
    }
}
```

(4) 在 src\fs. audioservice_test 包下新建一个 MyAudio. java 文件，在该文件中实现对名为"MY_AUDIO_SERVICE"的动作进行处理。代码为：

```
package fs.audioservice_test;
import java.io.IOException;
```

```
import android.app.Service;
import android.content.Intent;
import android.media.MediaPlayer;
import android.os.IBinder;
public class MyAudio  extends Service{
    private MediaPlayer mediaplayer;
     public IBinder onBind1(Intent arg0){
         //TODO 自动存根法
         return null;
     }
     @Override
     public void onDestroy(){
         //TODO 自动存根法
         super.onDestroy();
         if(mediaplayer!= null){
             mediaplayer.release();
             mediaplayer = null;
         }
     }
     @Override
     public int onStartCommand(Intent intent,int flags,int startId){
         //TODO 自动存根法
         super.onStartCommand(intent,flags,startId);
         mediaplayer = new MediaPlayer();
         try{
             String path = null;
            mediaplayer.setDataSource(path);
             mediaplayer.prepare();
             mediaplayer.start();
         }catch(IOException e){
             //TODO 自动存根法
             e.printStackTrace();
         }
        return startId;
     }
    @Override
    public IBinder onBind(Intent arg0) {
        //TODO 自动生成的方法存根
        return null;
    }
}
```

(5) 打开 AndroidManifest.xml 文件，在该文件中实现启动自定义的服务 MY_AUDIO_SERVICE 及关闭 Activity.java 文件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.audioservice_test"
    android:versionCode = "1"
    android:versionName = "1.0" >
    < uses - sdk
```

```
            android:minSdkVersion = "8"
            android:targetSdkVersion = "18" />
        < application
            android:allowBackup = "true"
            android:icon = "@drawable/ic_launcher"
            android:label = "@string/app_name"
            android:theme = "@style/AppTheme" >
            < activity
                android:name = "fs.audioservice_test.MainActivity"
                android:label = "@string/app_name" >
                < intent - filter >
                    < action android:name = "android.intent.action.MAIN" />
                      < category android:name = "android.intent.category.LAUNCHER" />
                </intent - filter >
                 </activity >
                 <! -- 调用自定义的 MY_AUDIO_SERVICE -- >
                < service android:name = "MyAudio">
                < intent - filter >
                    <! -- 关闭 Activity -- >
                      < action android:name = "fs.audioservice_test.MainActivity.MY_AUDIO_SERVICE" />
                    < category android:name = "android.intent.category.DEFAULT" />
                </intent - filter >
          </service >
        </application >
    </manifest >
```

运行程序，效果如图 9-3 所示。该服务启动 Mediaplayer，并播放存放在 SD 卡中的“sdcard/music/kaka.mp3”。



图 9-3　后台播放音频

9.1.3 Android 声音录制

Android SDK 提供了使用 MediaRecorder 类实现对音频和视频进行录制的功能。MediaRecorder 对象在运行过程中存在多种状态，其状态转化如图 9-4 所示。

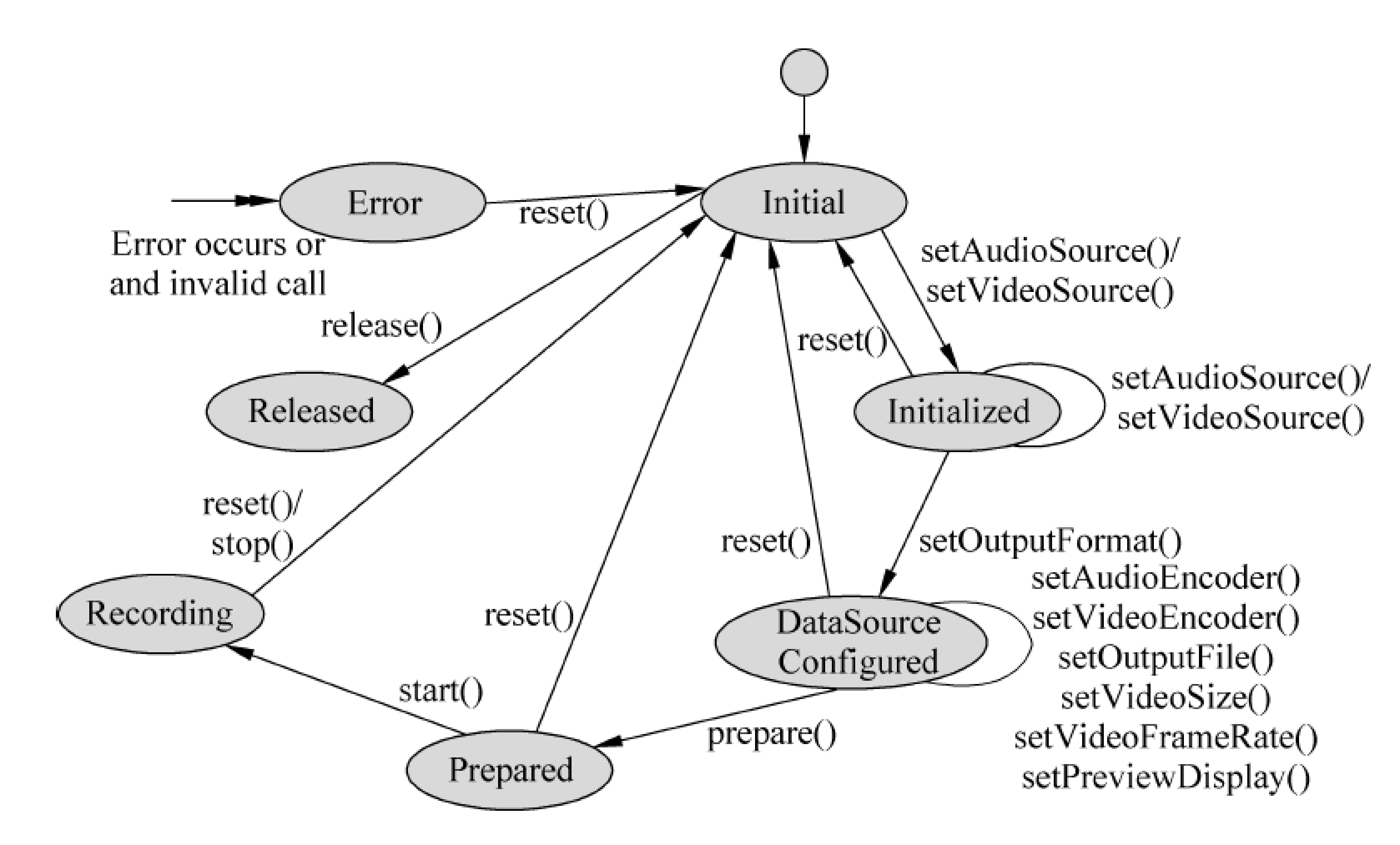


图 9-4 MediaRecorder 对象状态转化图

从图 9-4 中可看出：

(1) 创建 MediaRecorder 对象后处于 Initial 状态，MediaRecorder 对象会占用硬件资源，因此在不再需要时，应用调用 release()方法销毁。在其他状态调用 reset()方法，可以使得 MediaRecorder 对象重新回到 Initial 状态，达到复用 MediaRecorder 对象的目的。

(2) 在 Initial 状态调用 setVideoSource()或者 setAudioSource()之后，MediaRecorder 将进入 Initialized 状态。对于音频录制，目前 OPhone 平台支持从麦克风或者电话两个音频原染缸数据。在 Initialized 状态的 MediaRecorder 还需要设置编码格式、文件数据路径、文件格式等信息，设置之后 MediaRecorder 进入到 DataSourceConfigured 状态。

(3) 在 DataSourceConfigured 状态调用 prepare()方法，MediaRecorder 对象将进入 Prepared 状态，录制前的状态准备就绪。

(4) 在 Prepared 状态调用 start()方法，MediaRecorder 进入 Recording 状态，声音录制可能只需要一段时间，这时 MediaRecorder 一直处于录制状态。

(5) 在 Recording 状态调用 stop()方法，MediaRecorder 将停止录制，并将录制内容输出到指定文件。

MediaRecorder 定义了两个内部接口 OnErrorListener 和 OnInfoListener 来监听录制过程中的错误信息。例如，当录制的时间长度达到了最大限制或录制文件的大小达到了最大文件限制时，系统会回调已经注册的 OnInfoListener 接口的 onInfo()方法。

利用 MediaRecorder 实现录制步骤主要有：

(1) 创建 MediaRecorder 对象。

(2) 调用 MediaRecorder 对象的 setAudioSource()方法设置声音来源，一般传入 MediaRecorder.AudioSource.MIC 参数指定录制来自麦克风的声音。

(3) 调用 MediaRecorder 对象的 setOutputFormat()设置所录制的音频文件的格式。

（4）调用 MediaRecorder 对象的 setAudioEncoder()、setAudioEncodingBitRate(int bitRate)、setAudioSamplingRate(int samplingRate)设置所录制的声音的编码格式、编码位率、采样率等这些参数将可以控制所录制的声音的品质、文件的大小。一般来说，声音品质越好，声音文件越大。

（5）调用 MediaRecorder 的 setOutputFile(String path)方法录制音频文件的保存位置。

（6）调用 MediaRecorder 的 prepare()方法准备录制。

（7）调用 MediaRecorder 对象的 start()方法开始录制。

（8）录制完成，调用 MediaRecorder 对象的 stop()方法停止录制，并调用 release()方法释放资源。

下面通过一个实例来演示 MediaRecorder 对象的用法。

【例 9-4】 实现声音的录制，其实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Voice_Record。

(2) 打开 res\layout 目录下的 main. xml 布局文件，在文件中声明两个 Button 控件及一个 TextView 控件。代码为：

```
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = ".MainActivity"
    android:background = "#ffcc66">
    <Button
        android:id = "@+id/start"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignParentLeft = "true"
        android:layout_alignParentTop = "true"
        android:layout_marginLeft = "18dp"
        android:layout_marginTop = "30dp"
        android:text = "开始" />
    <Button
        android:id = "@+id/stop"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignBaseline = "@+id/start"
        android:layout_alignBottom = "@+id/start"
        android:layout_centerHorizontal = "true"
        android:text = "停止" />
    <TextView
        android:id = "@+id/textView1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_above = "@+id/stop"
        android:layout_alignParentLeft = "true"
        android:layout_alignParentTop = "true"
```

```
        android:layout_marginBottom = "14dp"
        android:text = "单击按钮开始录制声音" />
</RelativeLayout >
```


(3) 打开 src\fs. voice_record 包下的 MainActivity. java 文件,在文件中实现声明的录制效果。代码为:

```
package fs.voice_record;
import java.io.IOException;
import android.app.Activity;
import android.graphics.PixelFormat;
import android.media.MediaRecorder;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;
public class MainActivity extends Activity {
    private Button button_start;
    private Button button_stop;
    private MediaRecorder recorder;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getWindow().setFormat(PixelFormat.TRANSLUCENT);          //让界面横屏
        requestWindowFeature(Window.FEATURE_NO_TITLE);           //去掉界面标题
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
        //重新设置界面大小
        setContentView(R.layout.main);
        init();
    }
    private void init() {
        button_start = (Button) this.findViewById(R.id.start);
        button_stop = (Button) this.findViewById(R.id.stop);
        button_stop.setOnClickListener(new AudioListerner());
        button_start.setOnClickListener(new AudioListerner());
    }
    class AudioListerner implements OnClickListener {
        @Override
        public void onClick(View v) {
            if (v == button_start) {
                initializeAudio();
            }
            if (v == button_stop) {
                recorder.stop();                                  //停止刻录
                recorder.release();                               //刻录完成一定要释放资源
            }
        }
        private void initializeAudio() {
            recorder = new MediaRecorder();                       //新建 MediaRecorder 对象
            recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
            //设置 MediaRecorder 的音频源为麦克风
            recorder.setOutputFormat(MediaRecorder.OutputFormat.RAW_AMR);
```

```
            //设置 MediaRecorder 录制的音频格式
            recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
            //设置 MediaRecorder 录制音频的编码为 amr
            recorder.setOutputFile("/sdcard/peipei.amr");
            //设置录制好的音频文件的保存路径
            try {
                recorder.prepare();                         //准备录制
                recorder.start();                           //开始录制
            } catch (IllegalStateException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

(4) 打开 AndroidManifest.xml 配置文件，为文件添加相应权限。代码为：

```
…
        </activity>
    </application>
 <!-- 联网权限 -->
 <uses-permission android:name = "android.permission.INTERNET" />
 <!-- 往 SDCard 写入数据权限 -->
 <uses-permission android:name = "android.permission.WRITE_EXTERNAL_STORAGE" />
 <!-- 录音权限 -->
 <uses-permission android:name = "android.permission.RECORD_AUDIO" />
 <!-- 在 SDCard 中设置创建与删除文件的权限 -->
 <uses-permission android:name = "android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
</manifest>
```

运行程序，效果如图 9-5 所示。



图 9-5　声音录制

9.1.4 Android视频

与音频播放相比,视频的播放需要使用视觉组件将影像显示出来。在Android SDK中提供了多种播放视频文件的方法。例如可以用VideoView或SurfaceView来播放视频,其中使用VideoView组件播放视频最为方便。

1. VideoView播放视频

要想使用VideoView组件播放视频,首先需要在布局文件中创建该组件,然后在Activity中获取该组件,并应用其setVideoPath()方法或setVideoURI()方法加载要播放的视频,最后调用VideoView组件提供的stop()和pause()方法来停止或暂停视频的播放。

VideoView组件支持的XML属性主要有:

- android:id:用于设置组件的ID。
- android:background:用于设置背景,可以设置背景图片,也可以设置背景颜色。
- android:layout_width:用于设置宽度。
- android:layout_height:用于设置高度。

在Android中还提供了一个可以与VideoView组件结合使用的MediaController组件。MediaController组件用于通过图形控制界面来控制视频的播放。

下面通过一个实例来演示VideoView播放视频。

【例9-5】 播放视频。其具体实现步骤为:

(1) 在Eclipse中创建一个Android应用项目,命名为VideoView_test。

(2) 打开res\layout目录下的main.xml布局文件,在文件中定义一个TextView控件、一个VideoView控件及3个Button控件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< AbsoluteLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "#ccddee" >
    < TextView
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "@string/hello_world" />
    < VideoView
        android:id = "@ + id/VideoView01"
        android:layout_width = "320px"
        android:layout_height = "240px" />
    < Button
        android:id = "@ + id/PlayButton"
        android:layout_width = "90dp"
        android:layout_height = "wrap_content"
        android:layout_x = "111dp"
        android:layout_y = "204dp"
        android:text = "播放" />
    < Button
        android:id = "@ + id/LoadButton"
```

```
        android:layout_width = "84dp"
        android:layout_height = "wrap_content"
        android:layout_x = "17dp"
        android:layout_y = "202dp"
        android:text = "装载" />
    < Button
        android:id = "@ + id/PauseButton"
        android:layout_width = "93dp"
        android:layout_height = "wrap_content"
        android:layout_x = "215dp"
        android:layout_y = "208dp"
        android:text = "暂停" />
</AbsoluteLayout >
```

(3) 打开 src\fs.videoview_test 包下的 MainActivity.java 文件，在文件中实现利用 VideoView 及 MediaController 控件播放视频。代码为：

```
package fs.videoview_test;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.MediaController;
import android.widget.VideoView;
public class MainActivity extends Activity
{
    /** 第 1 次调用 activity 活动 */
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final VideoView videoView = (VideoView) findViewById(R.id.VideoView01);
        Button PauseButton = (Button) this.findViewById(R.id.PauseButton);
        Button LoadButton = (Button) this.findViewById(R.id.LoadButton);
        Button PlayButton = (Button) this.findViewById(R.id.PlayButton);
        //载入
        LoadButton.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0)
            {
                videoView.setVideoPath("/sdcard/bady.mp4");
                videoView.setMediaController(new MediaController(MainActivity.this));
                videoView.requestFocus();
            }
        });
        //播放
        PlayButton.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0)
            {
                videoView.start();
            }
```

```
            });
            //暂停
            PauseButton.setOnClickListener(new OnClickListener() {
                public void onClick1(View arg0)
                {
                    videoView.pause();
                }
                @Override
                public void onClick(View arg0) {
                    //TODO 自动生成的方法存根
                }
            });
        }
    }
```

运行程序，实现利用 VideoPlay 控件播放视频的效果如图 9-6 所示。



图 9-6　VideoPlay 播放视频

2. SurfaceView 播放视频

在前面介绍的 MediaPlayer 播放音频，实际上，MediaPlayer 还可以用来播放视频文件，只不过使用 MediaPlayer 播放视频时，没有提供图像输出界面。这时，可以使用 SurfaceView 组件来显示视频图像。使用 MediaPlayer 和 SurfaceView 来播放视频，大致可以分为 4 个步骤：

(1) 定义 SurfaceView 组件

定义 SurfaceView 组件可以在布局管理器中实现，也可以直接在 Java 代码中创建，不过推荐使用在布局器中实现。其基本格式为：

```
<SurfaceView
        android:id = "@ + id/surfaceView1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_alignLeft = "@ + id/textView1"
        android:layout_centerVertical = "true"
        android:layout_marginLeft = "36dp"
        android:background = "#ccddee"
        android:keepScreenOn = "true"/>
```

其中,android:keepScreenOn 属性用于指定在播放视频时决定是否需要打开屏幕。

(2) 创建 MediaPlayer 对象,并为其加载要播放的视频

与播放音频时创建 MediaPlayer 对象一样,也可以使用 MediaPlayer 类的静态方法 create()和无参的构造方法这两种方式创建 MediaPlayer 对象。

(3) 将所播放的视频画面输出到 SurfaceView

使用 MediaPlayer 对象的 setDisplay()方法可以将所播放的视频画面输出到 SurfaceView。setDisplay()方法的格式为:

```
setDisplay(SurfaceHolder sh)
```

参数 sh 用于指定 SurfaceHolder 对象,可以通过 SurfaceView 对象的 getHolder()方法获得。

(4) 调用 MediaPlayer 对象的相应方法控制视频的播放

使用 MediaPlayer 对象提供的 play()、pause()和 stop()方法,可以控制视频的播放、暂停和停止。

下面通过一个实例来演示怎样使用 MediaPlayer 和 SurfaceView 播放视频。

【例 9-6】 利用 SurfaceView 播放视频。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 SurfaceView_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件,在文件中声明一个 SurfaceView 控件、一个 TextView 控件、一个 SeekBar 控件、3 个 Button 控件及一个 MediaController 控件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#ffcc66">
   <SurfaceView
      android:id = "@ + id/VideoView01"
      android:layout_width = "320dip"
      android:layout_height = "240dip">
   </SurfaceView>
   <TextView
      android:id = "@ + id/TextView01"
      android:layout_width = "wrap_content"
      android:layout_height = "wrap_content"
      android:textColor = "#000000"
      android:textSize = "20dip"
```

```
        android:text = "0m:0s">
    </TextView>
    <SeekBar
        android:id = "@ + id/SeekBar01"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content">
    </SeekBar>
    <Button
            android:id = "@ + id/button1"
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
            android:text = "播放" />
    <Button
            android:id = "@ + id/button2"
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
            android:text = "暂停" />
    <Button
            android:id = "@ + id/button3"
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
            android:text = "停止" />
    <MediaController
        android:id = "@ + id/MediaController01"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"/>
    <LinearLayout
        android:id = "@ + id/LinearLayout01"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:orientation = "horizontal">
    </LinearLayout>
</LinearLayout>
```

(3) 打开 src\fs.surfaceview_test 包下的 MainActivity.java 文件，在文件中实现视频的播放。代码为：

```
package fs. surfaceview_test;
import android.app.Activity;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnCompletionListener;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.SeekBar.OnSeekBarChangeListener;
```

```
public class MainActivity extends Activity
implements OnClickListener,OnSeekBarChangeListener
{
    public static final int UPDATE_TIME = 0;                    //更新播放时间的消息编号
    public static final String currentPlay = "/sdcard/ * .3gp"; //播放路径
    Button play;                                                //“播放”按钮
    Button pause;                                               //“暂停”按钮
    Button stop;                                                //“停止”按钮
    SurfaceView sv;                                             //播放显示用的 SurfaceView
    SurfaceHolder sh;                                           //播放用的 SurfaceHolder
    MediaPlayer mp;                                             //媒体播放器
    SeekBar sb;                                                 //进度显示拖动条
    TextView tvTime;                                            //时间长度显示
    Handler hd;                                                 //消息处理器
    int state = 0;                    //播放状态指示: 0 - 未准备、1 - 播放中、2 - 暂停中
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //为“播放”、“暂停”、“停止”3 个按钮添加监听器
        play = (Button)this.findViewById(R.id.button1);
        pause = (Button)this.findViewById(R.id.button2);
        stop = (Button)this.findViewById(R.id.button3);
        play.setOnClickListener(this);
        pause.setOnClickListener(this);
        stop.setOnClickListener(this);
        //初始化播放用的 SurfaceView 及 SurfaceHolder
        sv = (SurfaceView)this.findViewById(R.id.VideoView01);
        sh = sv.getHolder();
        sh.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
        //初始化进度拖动条引用
        sb = (SeekBar)this.findViewById(R.id.SeekBar01);
        //不在暂停状态禁用拖拉条
        sb.setEnabled(false);
        //给进度拖动条添加监听器
        sb.setOnSeekBarChangeListener(this);
        //初始化显示播放时长的文本框
        tvTime = (TextView)findViewById(R.id.TextView01);
        //线程中创建一个 Handler
        hd = new Handler()
        {
            @Override
            public void handleMessage(Message msg)
            {
                //调用父类处理
                super.handleMessage(msg);
                //根据消息 what 编号的不同,执行不同的业务逻辑
                switch(msg.what)
                {
                  //将消息中的内容提取出来显示在 Toast 中
                  case UPDATE_TIME:
                      //获取消息中的数据
                      Bundle b = msg.getData();
```

```
                    //获取内容字符串
                    String msgStr = b.getString("msg");
                    //设置字符串到显示录音时长的文本框中
                    tvTime.setText(msgStr);
                break;
            }
        }
    };
}
public void onClick(View v) {
    if(v == play)
    {                                   //按下“播放”按钮
        if(state == 1)
        {                               //若当前已经是播放状态则报错返回
            Toast.makeText
            (
                this,
                "播放中,请结束本次播放再开始新的播放!",
                Toast.LENGTH_SHORT
            ).show();
            return;
        }
        else if(state == 0)
        {                               //若当前是未准备状态则进行播放前的准备工作
            //创建媒体播放器对象
            mp = new MediaPlayer();
            //设置音频流格式
            mp.setAudioStreamType(AudioManager.STREAM_MUSIC);
            //设置播放显示用 SurfaceView 的 SurfaceHolder
            mp.setDisplay(sh);
            //给视频播放结束事件添加的监听器
            mp.setOnCompletionListener(
                    new OnCompletionListener()
                    {
                        public void onCompletion(MediaPlayer arg0)
                        {           //歌曲播放结束停止播放并更新界面状态
                    state = 2;
                        }
                    }
            );
            try
            {
                //设置播放路径
                mp.setDataSource(currentPlay);
                //准备播放
                mp.prepare();
            }
            catch(Exception e)
            {
                e.printStackTrace();
            }
            //根据片长设置拖动条的最大值
            sb.setMax(mp.getDuration()/1000);
        }
```

```
            mp.start();                  //开始播放
            state = 1;                   //状态设置为 1 - 播放中
            sb.setEnabled(false);        //禁用拖动条
            new Thread()                 //启动一个线程定时更新进度条及时间
            {
                public void run()
                {
                    while(state == 1)
                    {
                        sb.setProgress(mp.getCurrentPosition()/1000);
                        setTime(mp.getCurrentPosition()/1000);
                        try
                        {
                            Thread.sleep(1000);
                        }
                        catch(Exception e)
                        {
                            e.printStackTrace();
                        }
                    }
                }
            }.start();
        }
        else if(v == pause)
        {                                //按下"暂停"按钮
            if(state!= 1)
            {                            //若当前不是播放状态
                Toast.makeText
                (
                    this,
                    "请在播放状态再暂停!",
                    Toast.LENGTH_SHORT
                ).show();
                return;
            }
            mp.pause();                  //暂停
            state = 2;                   //设置状态为 2 - 暂停
            sb.setEnabled(true);         //启用拖动条
        }
        else if(v == stop)
        {                                //按下"停止"按钮
            if(state == 0)
            {
                return;
            }
            state = 0;
            mp.stop();                   //停止播放
            mp.release();                //释放播放器
            mp = null;                   //清空引用
            sb.setEnabled(false);        //禁用拖动条
            sb.setProgress(0);           //设置进度为 0
            setTime(0);                  //设置时间为 0
        }
    }
    //设置显示时间的方法
    public void setTime(int countSecond)
    {
```

```
            //计算分钟和秒
            int second = countSecond % 60;
            int minute = countSecond/60;
            //创建内容字符串
            String msgStr = minute + "m:" + second + "s";
            //创建消息数据 Bundle
            Bundle b = new Bundle();
            //将内容字符串放进数据 Bundle 中
            b.putString("msg",msgStr);
            //创建消息对象
            Message msg = new Message();
            //设置数据 Bundle 到消息中
            msg.setData(b);
            //设置消息的 what 值
            msg.what = UPDATE_TIME;
            //发送消息
            hd.sendMessage(msg);
        }
        //实现进度拖拉的监听方法
        public void onProgressChanged(SeekBar seekBar, int progress,
                boolean fromUser) {
            if(mp!= null&&state == 2)
            {
                //进度拖拉到指定位置
                mp.seekTo(progress * mp.getDuration()/sb.getMax());
            }
        }
        public void onStartTrackingTouch(SeekBar seekBar) {     }
        public void onStopTrackingTouch(SeekBar seekBar) { }
    }
```

运行程序,效果如图 9-7 所示。



图 9-7　视频播放

9.1.5 Android 相机

照相机已成为手机的一个标配。Android 手机同样提供了完善的照相机功能。Android 系统在 android.hardware 软件包中提供了与照相机相关的一些类和方法。通过这些方法,应用程序可以完成拍照、录像、获取相机参数和修改设置相机参数等操作。

android.hardware 包中的 Camera 类没有构造方法,可以通过其提供的 open()方法打开相机。打开相机后,可以通过 Camera.Parameters 类处理相机的拍照参数。拍照参数设置完成后,可以调用 startPreview()方法预览拍照画面,也可以调用 takePicture()方法进行拍照。结束程序时,可以调用 Camera 类的 stopPreview()方法结束预览,并调用 Camera 类的 release()方法释放相机资源。Camera 类常用的方法有:

- getParameters():用于获取相机参数。
- Camera.open():用于打开相机。
- release():用于释放相机资源。
- setParameters(Camera.Parameters params):用于设置相机的拍照参数。
- setPreviewDisplay(SurfaceHolder holder):用于为相机指定一个用来显示相机预览画面的 SurfaceView。
- startPreview():用于开始预览画面。
- takePicture(Camera.ShutterCallback shutter,Camera.PictureCallback,Camera.PictureCallback.jpeg):用于进行拍照。
- stopPreview():用于停止预览画面。

应用创建自定义的相机,一般步骤如下:

(1) 检测相机硬件并获取访问。

(2) 建立一个 Preview 类:需要一个相机预览的类,继承 SurfaceView 类,并实现 SurfaceHolder 接口。

(3) 建立预览的布局。

(4) 为拍照建立监听。

(5) 拍照并且存储文件。

(6) 释放相机。

下面通过一个实例来演示手机相机的实现。

【例 9-7】 利用 Camera 类实现一个自定义相机。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 Camera_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件,在文件中声明一个 Button 控件,用于跳转到拍照界面。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "#aabbcc">
    <TextView
```

```
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:text = "@string/hello_world" />
        <Button
            android:id = "@ + id/camera_button"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_gravity = "center_horizontal"
            android:text = "拍照" />
</LinearLayout>
```

(3) 在 res\layout 目录下新建一个 camera.xml 布局文件，在文件中声明一个 SurfaceView 控件及一个 Button 控件。代码为：

```
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@ + id/linearLayout1"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical"
    android:background = "#cceeff">
    <SurfaceView
        android:id = "@ + id/surface_camera"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:layout_weight = "1"></SurfaceView>
    <Button
        android:id = "@ + id/take"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_gravity = "center_horizontal"
        android:text = "拍照" />
</LinearLayout>
```

(4) 打开 src\fs.camera_test 包下的 MainActivity.java 文件，在主界面中只需要完成跳转功能。代码为：

```
package fs.camera_test;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //TODO 自动存根法
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //实例化 Button 组件对象
        Button button  =  (Button) findViewById(R.id.camera_button);
    button.setOnClickListener(new Button.OnClickListener() {//为 Button 添加单击监听
```

```
                    @Override
                    public void onClick(View arg0) {
                        Intent intent = new Intent();      //初始化 Intent
                    //指定 intent 对象启动的类
                        intent.setClass(MainActivity.this,IamgeActivity.class);
                        startActivity(intent);             //启动新的 Activity
                    }
                });
        }
    }
```

(5) 在 src\fs.camera_test 包下新建一个 IamgeActivity.java 文件,主要用于实现图像预览过程。代码为:

```
package fs.camera_test;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.sql.Date;
import java.util.Timer;
import java.util.TimerTask;
import android.app.Activity;
import android.content.pm.ActivityInfo;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.PixelFormat;
import android.hardware.Camera;
import android.hardware.Camera.PictureCallback;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.os.Message;
import android.text.format.DateFormat;
import android.view.KeyEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;
public class IamgeActivity extends Activity implements SurfaceHolder.Callback {
    private Button btn_take;
    private SurfaceView surfaceView = null;            //创建一个 SurfaceView 组件对象
    private SurfaceHolder surfaceHolder = null;        //创建一个空 SurfaceHolder 对象
    private Camera camera = null;                      //创建一个空 Camera 对象
    private boolean previewRunning = false;            //预览状态
    /** 第 1 次调用 activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
            super.onCreate(savedInstanceState);
            getWindow().setFormat(PixelFormat.TRANSLUCENT);//窗口设置为半透明
            requestWindowFeature(Window.FEATURE_NO_TITLE); //窗口去掉标题
            getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
        WindowManager.LayoutParams.FLAG_FULLSCREEN);        //窗口设置为全屏
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
    setContentView(R.layout.camera);             //调用 setRequestedOrientation 来翻转 Preview
            //实例化 SurfaceView 对象
            surfaceView = (SurfaceView) findViewById(R.id.surface_camera);
              surfaceHolder = surfaceView.getHolder();   //获取 SurfaceHolder
            surfaceHolder.addCallback(this);             //注册实现正常的 Callback
            //设置缓存类型
            surfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
            btn_take = (Button)findViewById(R.id.take);
            btn_take.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                    //TODO Auto - generated method stub
                    if (camera != null) {                //判断 Camera 对象是否不为空
        //当按下"相机"按钮时,执行相机对象的 takePicture()方法,该方法有 3 个回调对象做入参,不
        //需要的时候可以设为 null
                        camera.takePicture(null,null,jpegCallback);
                        changeByTime(5000);              //调用延迟方法,5 秒后重新预览拍照
                    }
                }
            });
        }
        @Override
        public boolean onKeyDown(int keyCode,KeyEvent event) {
            //判断手机键盘按下的是否是拍照键、轨迹球键
            if (keyCode == KeyEvent.KEYCODE_CAMERA
                    || keyCode == KeyEvent.KEYCODE_DPAD_CENTER) {
                if (camera != null) {                    //判断 Camera 对象是否不为空
        //当按下"相机"按钮时,执行相机对象的 takePicture()方法,该方法有 3 个回调对象做入参,不
        //需要的时候可以设为 null
                    camera.takePicture(null,null,jpegCallback);
                    changeByTime(5000);                  //调用延迟方法,5 秒后重新预览拍照
                }
            }
            return super.onKeyDown(keyCode,event);
        }
        /**
          * 延迟方法
          * time 为毫秒
          */
        public void changeByTime(long time) {
            final Timer timer = new Timer();             //实例化 Timer 对象
            final Handler handler = new Handler() {
                @Override
                public void handleMessage(Message msg) {
                    switch (msg.what) {
```

```
                case 1:
                    stopCamera();                          //调用停止 Camera 方法
                    prepareCamera();                       //调用初始化 Camera 方法
                    startCamera();                         //调用开始 Camera 方法
                    timer.cancel();                        //撤销计时器
                    break;
                }
                super.handleMessage(msg);
            }
        };
        TimerTask task = new TimerTask() {
            @Override
            public void run() {
                Message message = new Message();
                message.what = 1;
                handler.sendMessage(message);
            }
        };
        timer.schedule(task,time);                         //设定运行任务的时间
    }
    /**
     * 当预览界面的格式和大小发生改变时,该方法被调用
     */
    @Override
    public void surfaceChanged(SurfaceHolder arg0,int arg1,int arg2,int arg3) {
        startCamera();                                     //调用开始 Camera 方法
    }
    /**
     * 初次实例化,预览界面被创建时,该方法被调用
     */
    @Override
    public void surfaceCreated(SurfaceHolder arg0) {
        prepareCamera();                                   //调用初始化 Camera 方法
    }
    /**
     * 当预览界面被关闭时,该方法被调用
     */
    @Override
    public void surfaceDestroyed(SurfaceHolder arg0) {
        stopCamera();                                      //调用停止 Camera 方法
    }
    /**
     * 初始化 Camera
     */
    public void prepareCamera() {
        camera = Camera.open();                            //初始化 Camera
        try {
            camera.setPreviewDisplay(surfaceHolder);       //设置预览
        } catch (IOException e) {
            camera.release();                              //释放相机资源
            camera = null;                                 //置空 Camera 对象
```

```
            }
        }
        /**
         * 开始 Camera
         */
        public void startCamera() {
            if (previewRunning) {                         //判断预览开启
                camera.stopPreview();                     //停止预览
            }
            try {
            Camera.Parameters parameters = camera.getParameters(); //获得相机参数对象
                parameters.setPictureFormat(PixelFormat.JPEG);     //设置格式
                //设置预览大小
                //parameters.setPreviewSize(480,320);
                //设置自动对焦
                //parameters.setFocusMode("auto");
                //设置图片保存时的分辨率大小
                //parameters.setPictureSize(2048,1536);
                camera.setParameters(parameters);            //给相机对象设置刚才设定的参数
               //设置用 SurfaceView 作为承载镜头取景画面的显示
                camera.setPreviewDisplay(surfaceHolder);
                camera.startPreview();                       //开始预览
                previewRunning = true;                       //设置预览状态为 true
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        /**
         * 停止 Camera
         */
        public void stopCamera() {
            if (camera != null) {                            //判断 Camera 对象不为空
                camera.stopPreview();                        //停止预览
                camera.release();                            //释放摄像头资源
                camera = null;                               //置空 Camera 对象
                previewRunning = false;                      //设置预览状态为 false
            }
        }
        //照片拍摄之后的事件
        private PictureCallback jpegCallback = new PictureCallback() {
            @Override
            public void onPictureTaken(byte[] arg0,Camera arg1) {
                //获取存储卡(SDCard)的根目录
                String sdCard = Environment.getExternalStorageDirectory().getPath();
                //获取相片存放位置的目录
                String dirFilePath = sdCard + File.separator + "Camera";
                //获取当前时间的自定义字符串
                String date = (String) DateFormat.format("yyyy-MM-dd hh-mm-ss",new java.
util.Date());
            //onPictureTaken 传入的第 1 个参数及为相片的 byte,实例化 Bitmap 对象
                Bitmap bitmap = BitmapFactory.decodeByteArray(arg0,0,arg0.length);
```

```
            try {
                File dirFile = new File(dirFilePath);       //创建相片存放位置的 File 对象
                if (!dirFile.exists()) {                    //判断路径是否不存在
                    dirFile.mkdir();                        //创建该文件夹
                }
                //创建 1 个前缀为 photo,扩展名为.jpg 的图片文件,
                //createTempFile 方法来创建是为了避免文件重复
                File file = File.createTempFile("img-",date + ".jpg",dirFile);
                BufferedOutputStream bOutputStream = new BufferedOutputStream(new
FileOutputStream(file));
                //采用压缩文件的方法
                bitmap.compress(Bitmap.CompressFormat.JPEG,80,bOutputStream);
                //清除缓存,更新 BufferedOutputStream
                bOutputStream.flush();
                //关闭 BufferedOutputStream
                bOutputStream.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    };
}
```

(6) 设置权限,打开 AndroidManifest.xml 文件,在文件中设置照相机权限。代码为:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="fs.camera_test"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.WRITE_SETTINGS" />
    <!-- 在 SDCard 中设置创建与删除文件的权限 -->
    <uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
    <!-- 往 SDCard 写入数据权限 -->
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

 <activity
            android:name="fs.camera_test.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
```

```
                </intent-filter>
            </activity>
            <!-- 权限 -->
                <activity
                android:name=".IamgeActivity"
                android:screenOrientation="portrait">
            </activity>
        </application>
    </manifest>
```

运行程序,效果如图 9-8 所示。



图 9-8　照相机功能

9.2　Android 无线网络

WiFi(WIreless Fidelity)是一种可以将个人计算机、手机设备(如 PDA、手机)等终端以无线方式相互连接的技术。WiFi 是由一个名为“无线以太网相容联盟”(Wireless Ethernet Compatibility Alliance,WECA)的组织所发布的业界术语,中文译为“无线相容认证”。

随着通信技术的发展以及 IEEE 802.11a 和 IEEE 8.2.11g 等标准的出现,现在 IEEE 802.11 标准已被统称作为 WiFi。1997 年 IEEE 802.11 第 1 个版本发表,其中定义了介质访问接入控制层(MAC 层)和物理层。物理层定义了工作在 2.4GHz 的 ISM 频段上的两种无线调频方式和一种红外传输的方式,总数据传输速率设计为 2Mbit/s。两个设备之间的通信可以自由直接(ad hoc)的方式进行,也可以在基站(Base Station,BS)或者访问点(Access Point,AP)的协调下进行。1999 年加上了两个补充版本:802.11a 定义了一个在

5GHz ISM 频段上的数据传输速率可达 54Mbit/s 的物理层，802.11b 定义了一个在 2.4GHz 的 ISM 频段上但数据传输速率高达 11Mbit/s 的物理层。WiFi 的正式名称是“IEEE 802.11b”。

WiFi 是一种帮助用户访问电子邮件、Web 和流式媒体的互联网技术，它为用户提供了无线的宽带互联网访问。同时，它也是在家里、办公室或在旅途中比较快速、便捷的上网途径。WiFi 在掌上设备应用越来越广泛，而智能手机就是其中一分子。与早期应用于手机上的蓝牙技术不同，WiFi 具有更大的覆盖范围和更高的传输速率，因此 WiFi 手机成为了目前移动通信业界的时尚潮流。

WiFi 无线连接的描述，包括接入点、网络连接状态、隐藏的接入点、IP 地址、连接速度、MAC 地址、网络 ID、信号强度等信息。其主要常用方法有：

- getBSSID()：获取 BSSID。
- getDetailedStateOf()：获取客户端的连通性。
- getHiddenSSID()：获得 SSID 是否被隐藏。
- getIpAddress()：获取 IP 地址。
- getLinkSpeed()：获得连接的速度。
- getMacAddress()：获得 Mac 地址。
- getRssi()：获得 802.11n 网络的信号。
- getSSID()：获得 SSID。
- getSupplicanState()：返回具体客户端状态的信息。

下面通过一个实例来演示 WiFi 开发。

【例 9-8】 使用 WiFi 可以进行连接设备搜索并获取相应信息的过程。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 WiFi_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明一个 ScrollView 控件，在控件中声明一个 LinearLayout 布局，在布局中声明一个 TextView 控件及 4 个 Button 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<ScrollView xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
 <LinearLayout
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#ccddee">
<Button
    android:id = "@ + id/scan"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "扫描网络" />
<Button
    android:id = "@ + id/start"
    android:layout_width = "wrap_content"
```

```
        android:layout_height = "wrap_content"
        android:text = "打开 WiFi"/>
    < Button
        android:id = "@ + id/stop"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "关闭 WiFi" />
    < Button
        android:id = "@ + id/check"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "WiFi 状态" />
     < TextView
        android:id = "@ + id/allNetWork"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "当前没有扫描到 WiFi 网络" />
        </LinearLayout >
    </ScrollView >
```

(3) 在 src\fs. wifi_test 包下新建一个 WiFi 的相关操作都封装在了一个 WiFiActivity 类中,以后开启或关闭等相关操作可以直接调用这个类的相关方法。代码为:

```
package fs.wifi_test;
import java.util.List;
import android.content.Context;
import android.net.wifi.ScanResult;
import android.net.wifi.WifiConfiguration;
import android.net.wifi.WifiInfo;
import android.net.wifi.WifiManager;
import android.net.wifi.WifiManager.WifiLock;
public class WiFiActivity {
    //定义一个 WifiManager 对象
    private WifiManager mWifiManager;
    //定义一个 WifiInfo 对象
    private WifiInfo mWifiInfo;
    //扫描出的网络连接列表
    private List < ScanResult > mWifiList;
    //网络连接列表
    private List < WifiConfiguration > mWifiConfigurations;
    WifiLock mWifiLock;
    public WiFiActivity(Context context){
        //取得 WifiManager 对象
        mWifiManager = (WifiManager) context.getSystemService(Context.WIFI_SERVICE);
        //取得 WifiInfo 对象
        mWifiInfo = mWifiManager.getConnectionInfo();
    }
    //打开 WiFi
    public void openWifi(){
        if(!mWifiManager.isWifiEnabled()){
            mWifiManager.setWifiEnabled(true);
```

```
        }
    }
    //关闭 WiFi
    public void closeWifi(){
        if(!mWifiManager.isWifiEnabled()){
            mWifiManager.setWifiEnabled(false);
        }
    }
    //检查当前 WiFi 状态
    public int checkState() {
        return mWifiManager.getWifiState();
    }
    //锁定 WifiLock
    public void acquireWifiLock(){
        mWifiLock.acquire();
    }
    //解锁 WifiLock
    public void releaseWifiLock(){
        //判断是否锁定
        if(mWifiLock.isHeld()){
            mWifiLock.acquire();
        }
    }
    //创建一个 WifiLock
    public void createWifiLock(){
        mWifiLock = mWifiManager.createWifiLock("test");
    }
    //得到配置好的网络
    public List<WifiConfiguration> getConfiguration(){
        return mWifiConfigurations;
    }
    //指定配置好的网络进行连接
    public void connetionConfiguration(int index){
        if(index > mWifiConfigurations.size()){
            return ;
        }
        //连接配置好指定 ID 的网络
        mWifiManager.enableNetwork(mWifiConfigurations.get(index).networkId,true);
    }
    public void startScan(){
        mWifiManager.startScan();
        //得到扫描结果
        mWifiList = mWifiManager.getScanResults();
        //得到配置好的网络连接
        mWifiConfigurations = mWifiManager.getConfiguredNetworks();
    }
    //得到网络列表
    public List<ScanResult> getWifiList(){
        return mWifiList;
    }
    //查看扫描结果
```

```
    public StringBuffer lookUpScan(){
        StringBuffer sb = new StringBuffer();
        for(int i = 0;i < mWifiList.size();i++){
            sb.append("Index_" + new Integer(i + 1).toString() + ":");
             //将 ScanResult 信息转换成一个字符串包
            //包括 BSSID、SSID、capabilities、frequency、level
            sb.append((mWifiList.get(i)).toString()).append("\n");
        }
        return sb;
    }
    public String getMacAddress(){
        return (mWifiInfo == null)?"NULL":mWifiInfo.getMacAddress();
    }
    public String getBSSID(){
        return (mWifiInfo == null)?"NULL":mWifiInfo.getBSSID();
    }
    public int getIpAddress(){
        return (mWifiInfo == null)?0:mWifiInfo.getIpAddress();
    }
    //得到连接的 ID
    public int getNetWordId(){
        return (mWifiInfo == null)?0:mWifiInfo.getNetworkId();
    }
    //得到 WifiInfo 的所有信息
    public String getWifiInfo(){
        return (mWifiInfo == null)?"NULL":mWifiInfo.toString();
    }
    //添加一个网络并连接
    public void addNetWork(WifiConfiguration configuration){
        int wcgId = mWifiManager.addNetwork(configuration);
        mWifiManager.enableNetwork(wcgId,true);
    }
    //断开指定 ID 的网络
    public void disConnectionWifi(int netId){
        mWifiManager.disableNetwork(netId);
        mWifiManager.disconnect();
    }
}
```

(4) 打开 src\fs.wifi_test 包的 MainActivity.java 文件,实现 WiFi 的相关操作。代码为:

```
package fs.wifi_test;
import java.util.List;
import android.app.Activity;
import android.net.wifi.ScanResult;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
```

```
public class MainActivity extends Activity {
    /** 第 1 次调用 activity 活动 */
     private TextView allNetWork;
        private Button scan;
        private Button start;
        private Button stop;
        private Button check;
        private WiFiActivity mwifiactivity;
        //扫描结果列表
        private List<ScanResult> list;
        private ScanResult mScanResult;
        private StringBuffer sb = new StringBuffer();
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mwifiactivity = new WiFiActivity(MainActivity.this);
        init();
    }
    public void init(){
        allNetWork = (TextView) findViewById(R.id.allNetWork);
        scan = (Button) findViewById(R.id.scan);
        start = (Button) findViewById(R.id.start);
        stop = (Button) findViewById(R.id.stop);
        check = (Button) findViewById(R.id.check);
        scan.setOnClickListener(new MyListener());
        start.setOnClickListener(new MyListener());
        stop.setOnClickListener(new MyListener());
        check.setOnClickListener(new MyListener());
    }
    private class MyListener implements OnClickListener{
         @Override
         public void onClick(View v) {
             //TODO:自动存根法
             switch (v.getId()) {
             case R.id.scan:                          //扫描网络
                  getAllNetWorkList();
                 break;
            case R.id.start:                          //打开 WiFi
               mwifiactivity.openWifi();
                  Toast.makeText(MainActivity.this,"当前 WiFi 状态为: " + mwifiactivity.
checkState(),1).show();
                 break;
            case R.id.stop:                           //关闭 WiFi
               mwifiactivity.closeWifi();
                  Toast.makeText(MainActivity.this,"当前 WiFi 状态为: " + mwifiactivity.
checkState(),1).show();
                 break;
            case R.id.check:                          //WiFi 状态
                 Toast.makeText(MainActivity.this,"当前 wifi 状态为: " + mwifiactivity.
checkState(),1).show();
```

```
                break;
            default:
                break;
            }
        }
    }
    public void getAllNetWorkList(){
          //每次单击扫描之前清空上一次的扫描结果
        if(sb!= null){
            sb = new StringBuffer();
        }
        //开始扫描网络
        mwifiactivity.startScan();
        list = mwifiactivity.getWifiList();
        if(list!= null){
            for(int i = 0;i < list.size();i++){
                //得到扫描结果
                mScanResult = list.get(i);
                sb = sb.append(mScanResult.BSSID + "   ").append(mScanResult.SSID + "   ")
                .append(mScanResult.capabilities + "   ").append(mScanResult.frequency + "   ")
                .append(mScanResult.level + "\n\n");
            }
            allNetWork.setText("扫描到的 WiFi 网络: \n" + sb.toString());
        }
    }
}
```

(5) 打开 AndroidManifest.xml 文件，实现 WiFi 开发的权限设置。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.wifi_test"
    android:versionCode = "1"
    android:versionName = "1.0" >
    < uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
   <! -- 以下是使用 WiFi 访问网络所需的权限 -- >
      < uses - permission android:name = "android.permission.CHANGE_NETWORK_STATE"/>
    < uses - permission android:name = "android.permission.CHANGE_WIFI_STATE"/>
    < uses - permission android:name = "android.permission.ACCESS_NETWORK_STATE"/>
    < uses - permission android:name = "android.permission.ACCESS_WIFI_STATE"/>
    < application
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        < activity
            android:name = "fs.wifi_test.MainActivity"
            android:label = "@string/app_name" >
            < intent - filter >
```

```
                <action android:name = "android.intent.action.MAIN" />
                <category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter>
        </activity>
    </application>
</manifest>
```

运行程序,效果如图 9-9 所示。



图 9-9　WiFi 的开发

9.3　Android 通信

智能手机作为目前 Android 系统的最大载体,Android 系统不仅需要满足用户的娱乐需求,更重要的也是必须解决的是手机最基本的两大需求——语音通话和短信。

9.3.1　Android 语音通话

语音通话是手机设备最基本的也是最重要的功能。在 Android 系统中不仅可以统计通话号码、通话时间等基本信息,还可以很方便地拨出号码、自动挂断或接通电话以及电话录音等。

1. Intent

Intent 被译为“意图”,在 Android 中提供了一 Intent 机制来协助应用间的交互与通信。Intent 负责对应用中一次操作的动作、动作涉及数据、附加数据进行描述,Android 则根据此 Intent 的描述,负责找到对应的组件,将 Intent 传递给调用的组件,并完成组件的调用。

Intent 不仅可用于应用程序之间，也可用于应用程序内部 Activity/Service 之间的交互。因此，可以将 Intent 理解为不同组件之间通信的“媒介”，专门提供组件互相调用的相关信息。

Intent 是对它要完成的动作的一种抽象描述，Intent 封装了它要执行动作的属性：Action（动作）、Data（数据）、Category（类别）、Type（类型）、Component（组件信息）和 Extras（附加信息）。

(1) Action

Action 是指 Intent 要实施的动作，是一个字符串常量。如果指明了一个 Action，执行者就会依照这个动作的指示，接收相关输入，表现对应行为，产生符合条件的输出。

在 Intent 类中，定义了大量的 Action 常量属性，主要有：

- ACTION_MAIN：作为一个主要的进入口，而并不期望去接收数据。
- ACTION_VIEW：向用户显示数据。
- ACTION_ATTACH_DATA：用于指定一些数据应该附属于哪些地方，例如，图片数据应该附属于联系人。
- ACTION_EDIT：访问已给的数据，提供明确的可编辑接口。
- ACTION_PICK：从数据中选择一个子项目，并返回所选中的项目。
- ACTION_CHOOSER：显示一个 activity 选择器，允许用户在进程之前选择他们想要的。
- ACTION_CET_CONTENT：允许用户选择特殊种类的数据，并返回（特殊种类的数据：照一张相片或录一段音）。
- ACTION_DIAL：拨打一个指定的号码，显示一个带有号码的用户界面，允许用户去启动呼叫。
- ACTION_CALL：根据指定的数据执行一次呼叫。
- ACTION_SEND：传递数据，被传送的数据没有指定。
- ACTION_SENDTO：发送一个信息到某个指定的人。
- ACTION_ANSWER：处理一个打进电话呼叫。
- ACTION_INSERT：插入一条空项目到已给的容器。
- ACTION_DELETE：从容器中删除已给的数据。
- ACTION_RUN：运行数据。
- ACTION_SYNC：同步执行一个数据。
- ACTION_PICK_ACTIVITY：为已知的 Intent 选择一个 Activity，返回被选中的类。
- ACTION_SEARCH：执行一次搜索。
- ACTION_WEB_SEARCH：执行一次 Web 搜索。
- ACTION_FACTORY_TEST：工厂测试的主要进入点。

(2) Data

Intent 的 Data 属性是执行动作的 URI 和 MIME 类型，不同的 Action 有不同的 Data 数据指定。例如，通讯录中 identifier 为 1 联系人的信息（一般以 U 形式被描述），给这个人打电话的语句为：

```
ACTION_VIEW content://contacts/1
```

```
ACTION_DIAL content://contacts/1
```

(3) Category

Intent 中的 Category 属性起着对 Action 补充说明的作用。通过 Action，配合 Data 或 Type 可以准确表达出一个完整的意图(加一些约束会更精准)。Intent 中的 Category 属性是一个执行 Action 的附加信息。例如，CATEGORY_LAUNCHER 表示加载程序时 Activity 出现在最上面；_HOME 表示回到 Home 界面。

(4) Type

Intent 的 Type 属性显示指定 Intent 的数据类型(MIME)。通常 Intent 的数据类型可以从 Data 自身判断出来，但是一旦指定了 Type 类型，就会强制使用 Type 指定的类型而不再进行推导。

(5) Component

Intent 的 Component 属性指定 Intent 的目标组件的类名称。通常情况下，Android 会根据 Intent 中包含的其他属性的信息进行查找，例如用 Action、Data、Type、Category 去描述一个请求，这种模式称为 Implicit Intents。通过这种模式，提供一种灵活可扩展的模式，给用户和第三方应用一个选择权。例如，一个邮箱软件，大部分功能都不错，但是选择图片的功能不尽如人意，如果采用 Implicit Intents，那么它就是一个开放的体系，如果想用手机中的其他图片代替邮箱中默认的，可以完成这一功能。但该模式需要付出性能上的开销，因为毕竟存在一个检索过程。于是 Android 提供了另一种模式 Explicit Intents，该模式需要 Component 对象。Component 就是完整的类名，形如 com. xxxxx. xxxx，一旦指明就可以直接调用。根据该属性是否被指定，Intent 可分为显式 Intent 和隐式 Intent。

(6) Extra

Intent 的 Extra 属性是添加一些组件的附加信息。例如，要通过一个 Activity 执行“发送电子邮件”这个动作请求，可以将电子邮件的 subject、body 等保存在 extras 里，传给电子邮件的发送组件。

(7) Flags

Flags 表示不同来源的标记，多数用于指示 Android 系统怎样启动 Activity(如 Activity 属于哪个 Task)以及启动后怎样对待(如它是否属于近期的 Activity 列表)。所有标记都定义在 Intent 类中。

说明：所有标记都是整数类型。

2. 显式 Intent 和隐式 Intent

Intent 有显式和隐式之分，显式的 Intent 是根据组件的名称直接启动要启动的组件，如 Service 或者 Activity；隐式的 Intent 通过配置的 Action、Category、Data 来找到匹配的组件并启动。

3. IntentFilter

为支持隐式 Intent，可以声明一个甚至多个 IntentFilter。每个 IntentFilter 描述组件所允许响应 Intent 请求的能力。例如请求网页浏览器，网页浏览器程序的 IntentFilter 就应该声明它所希望接收的 IntentFilter Action 是 WEB_SEARCH_ACTION 以及与之相关的请求数据是网页地址 URI 格式。

如何为组件声明自己的 IntentFilter 呢？常见的方法是在 Android Manifest. xml 文件

中用属性<Intent-Filter>描述组件的 IntentFilter。

一个隐式 Intent 请求要能够传递目标组件，必须通过 Action、Data 以及 Category 这 3 个方面的检查。如果任何一方面不匹配，Android 都不会将该隐式 Intent 传递给目标组件。这 3 方面检查的规则主要为：

(1) 动作测试

<intent-Filter>元素中可以包含子元素<action>，例如：

```
<intent-Filter>
<action android:name = "com.example.project.SHOW-CURRENT"/>
<action android:name = "com.example.project.SHOW-RECENT"/>
<action android:name = "com.example.project.SHOW-PENDING"/>
</intent-Filter>
```

一条<intent-Filter>元素至少包含一个<action>，否则任何 Intent 请求都不能和该<intent-Filter>匹配。

如果 Intent 请求的 Action 和<intent-Filter>中的某一条<action>匹配，那么该 Intent 就通过了这条<intent-Filter>的动作测试。

如果 Intent 请求或<intent-Filter>中没有说明具体的 Action 类型，那么就会出现下面这两种情况：

- 如果<intent-Filter>中没有包含任何 Action 类型，那么无论什么 Intent 请求都无法和这条<intent-Filter>匹配。
- 反之，如果 Intent 请求中没有设定 Action 类型，那么只要<intent-Filter>中包含有 Action 类型，这个 Intent 请求就顺利通过<intent-Filter>的行为测试。

(2) 类别测试

<intent-Filter>元素可以包含<category>子元素，例如：

```
<intent-Filter>
<category android:name = "android.Intent.Category.DEFALT"/>
<category android:name = "android.Intent.Category.BROWSABLE"/>
</intent-Filter>
```

只有当 Intent 请求中所有的 Category 与组件中某一个 Intent 的<category>完全匹配时，才会让该 Intent 请求通过测试，IntentFilter 中多余的<category>声明并不会导致匹配失败。一个没有指定任何类别测试的 IntentFilter 仅仅匹配没有设置类别的 Intent 请求。

(3) 数据测试

数据在<intent-Filter>的描述为：

```
<intent-Filter>
<data android:type = "video/mpeg" android:scheme = "http" … />
<data android:type = "audio/mpeg" android:scheme = "http" … />
</intent-Filter>
```

<data>元素指定了要接收的 Intent 请求的数据 URI 及数据类型，其中 URI 被分成 3 部分来进行匹配：scheme、authority 和 path。用 setData()设定的 Intent 请求的 URI 数据类型和 scheme 必须与 IntentFilter 中所指定的一致。如果 IntentFilter 中还指定了

authority 或 path，它们也需要匹配才会通过测试。

4. 拨打电话

借助于 Intent，可以轻松地实现拨打电话的应用程序。只需声明一个拨号的 intent 对象，并使用 startActivity()方法启动即可。

创建 Intent 对象的代码为 Intent intent＝new Intent(action，uri)，其中 uri 是要拨叫的号码数据，通过 Uri. parse()方法把"tel：1234"格式的字符串转换为 URI。而 action 有两种使用方式：一种是 Intent. Action_CALL，直接进行呼叫方式，这种方式需要应用程序具有 android：permission. CALL_PHONE 权限。另外一种是 Intent. Action_DIAL，这种方式不直接进行呼叫，而是启动 Android 系统的拨号应用程序，然后由用户进行拨号。这种方式不需要任何权限的设置。

下面通过一个实例演示怎样使用 Intent 实现电话拨打。

【例 9-9】 使用 Intent 拨打电话。其实现具体步骤为：

(1) 在 Eclipse 中创建 Android 应用项目，命名为 Phone_Call，实现拨打电话功能。

(2) 在 res\layout 文件夹中打开布局文件 main. xml。在文件中添加一个编辑框和一个按钮，并修改其默认属性。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "@drawable/kp"
    android:orientation = "vertical" >
    <EditText
        android:id = "@ + id/editText1"
        android:layout_width = "276dp"
        android:layout_height = "wrap_content"
        android:layout_alignParentTop = "true"
        android:layout_centerHorizontal = "true"
        android:layout_gravity = "center_vertical"
        android:layout_marginTop = "42dp"
        android:ems = "10"
        android:inputType = "phone" />
    <Button
        android:id = "@ + id/button1"
        android:layout_width = "123dp"
        android:layout_height = "79dp"
        android:layout_gravity = "center_horizontal"
        android:layout_marginTop = "85dp"
        android:text = "拨打"
        android:textColor = "@android:color/black" />
</LinearLayout>
```

(3) 在 src\fs. phone_call 文件中，打开 MainActivity 文件并编写，它从页面中获得用户输入的电话号码。通过为按钮增加单击事件监听器来完成拨号功能。代码为：

```
package fs.phone_call;
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.app.Activity;
import android.view.Menu;
public class MainActivity extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);                          //设置页面布局
        EditText numberTV = (EditText) findViewById(R.id.editText1);
        //通过 ID 值获得文本框对象
        final String number = numberTV.getText().toString();  //获得输入的电话号码
        Button dial = (Button) findViewById(R.id.button1);     //通过 ID 值获得按钮对象
        dial.setOnClickListener(new View.OnClickListener()
        {
            public void onClick(View v)
            {
                Intent intent = new Intent();                   //创建 Intent 对象
                intent.setAction(Intent.ACTION_DIAL);           //设置 Intent 动作
                intent.setData(Uri.parse("tel:" + number));     //设置 Intent 数据
                startActivity(intent);                          //将 Intent 传递给 Activity
            }
        });
    }
}
```

(4) 修改 AndroidMainfest.xml 文件，增加拨打电话的权限。代码为：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="fs.phone_call"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <Activity
            android:name=".MainActivity"
```

```
            android:label = "@string/app_name" >
            <intent - filter >
                <action android:name = "android.intent.action.MAIN" />
                <category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter >
        </Activity >
    </application >
  <uses - permission android:name = "android.permission.CALL_PHONE"/>
</manifest >
```

运行程序，效果如图 9-10 所示。在编辑框中输入需要拨打的电话，单击“拨打”按钮即可完成拨号功能。



图 9-10　拨打电话界面

5. 来电防火墙

有呼出就有对应的呼入电话。有时有不想接的电话，在 Android 中可以实现拦截。在 Android 中提供了 TelephonyManager 类用于实现对象获取。

(1) 获取 TelephonyManager 对象。TelephonyManager 是系统的服务，获取该对象，实现为：

```
TelephonyManager tm = (TelephonyManager)Content.getSystemService(Context.TELEPHONY_SERVICE)
```

(2) 显示方式。在 TelephonyManager 中提供了很多有用的方法，可以获取当前电话状态、SIM 卡信息、电话网络状态等信息。常用方法为：

- int getCallState()：获取当前电话的状态，返回值 CALL_STATE_IDLE，表示电话无活动，即电话已经被挂断；CALL_STATE_OFFHOOK，表示摘机，即电话正在通话中；CALL_STATE_RINGING，表示响铃，即有电话正在呼入。当接收到广播

时,就根据这些不同的电话状态进行相应的处理。

- int getSimState():获取当前手机中的 SIM 卡的状态,常用的返回值有 SIM_STATE_READY,表示 SIM 卡可用、状态良好;SIM_STATE_ABSENT,表示没有 SIM 卡或当前 SIM 不可用。
- String getSimSerialNumber():获取 SIM 卡号。
- String getSimOperator():获取 SIM 卡的提供商代码。代码由国家编号和网络标号 MCC+MNC(Mobile Country Code+Mobile Network Code)共同组成。
- int getNetWorkType():获取手机类型,返回值有 PHONE_TYPE_NONE,无信号;PHONE_TYPE_GSM,表示 GSM 信号;PHONE_TYPE_CDMA,表示 CDMA 信号。
- int getNetWorkType():获取当前使用的网络类型。返回值包括了全球主要的网络类型,在国内常使用到的有 NETWORK_TYPE_UNKNOWN,表示网络类型未知类型;NETWORK_TYPE_GPRS,表示 GPRS 网络;NETWORK_TYPE_EDGE,表示 EDGE 网络;NETWORK_TYPE_UMTS,表示 UMTS 网络。
- String getDeviceId():获取设备的唯一表示 ID,即 GSM 手机的 IMEI 码或 CDMA 手机的 MEID 码。

接着利用一个实例来演示 TelephonyManager 类的使用。

【例 9-10】 利用 TelephonyManager 类实现来电防火墙功用。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,名为 Telephony_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件。文件代码为:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity"
    android:background="#cceeff">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
</RelativeLayout>
```

(3) 打开 src\fs.telephony_test 包下的 MainActivity.java 文件,用于实现跳转到主界面。代码为:

```
package fs.telephony_test;
import android.app.Activity;
import android.os.Bundle;
public class MainActivity extends Activity {
    /** 第 1 次调用 activity 活动 */
```

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

(4) 在 src\fs.telephony_test 包下新建一个 TelephoneActivity.java 文件,该文件用于实现来电防火墙。代码为:

```
package fs.telephony_test;
import java.util.Timer;
import java.util.TimerTask;
import android.app.Service;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.telephony.TelephonyManager;
import android.util.Log;
import android.widget.Toast;
public class TelephoneActivity extends BroadcastReceiver {
    Context mcontext;
    String TAG = "CALL";
    String phoneNumber = null;
    //返回到主界面
    TimerTask task = new TimerTask() {
        public void run() {
            Intent i = new Intent(Intent.ACTION_MAIN);
            i.addCategory(Intent.CATEGORY_HOME);
            i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            mcontext.startActivity(i);
            Log.i(TAG,"task start");
        }
    };
    @Override
    public void onReceive(Context context,Intent intent) {
        //TODO 自动存根法
        mcontext = context;
        TelephonyManager tm = (TelephonyManager) context
                .getSystemService(Service.TELEPHONY_SERVICE);
        switch (tm.getCallState()) {
        case TelephonyManager.CALL_STATE_RINGING:              //来电响铃
            Log.i(TAG,"CALL_STATE_RINGING");
            try {
                //来电拒听
                phoneNumber = intent.getStringExtra("incoming_number");
                Log.i(TAG,"call number is " + phoneNumber);
                if (phoneNumber.equals("10086")) {
                    UTL.getITelephony(tm).endCall();
                Toast.makeText(context,"号码" + phoneNumber + "已经被挂断拦截",1000).show();
                    Log.i(TAG,"号码" + phoneNumber + "已经被挂断拦截");
```

```
                }
                //静默接通电话
              //timer.schedule(task,300);
            } catch (Exception e) {
                Log.i(TAG,"ring w " + e.toString());
            }
            break;
        case TelephonyManager.CALL_STATE_OFFHOOK:            //来电接通,去电拨出
            Log.i(TAG,"CALL_STATE_OFFHOOK");
            break;
        case TelephonyManager.CALL_STATE_IDLE:               //来去电电话挂断
            Log.i(TAG,"CALL_STATE_IDLE");
            break;
        }
    }
}
```

(5) 在 src\fs.telephony_test 包下新建一个 UTL.java 文件,该文件使用 Java 的反射机制,从公开的 TelephonyManager 中实例化出添加的源码包 com.android.internal.telephony 中的 ITelephony 接口。

```
package fs.telephony_test;
import java.lang.reflect.Method;
import android.telephony.TelephonyManager;
public class UTL {
    /**
      * 从 TelephonyManager 中实例化 ITelephony,并返回
      */
    static public com.android.internal.telephony.ITelephony getITelephony(TelephonyManager
telManager) throws Exception {
        Method getITelephonyMethod = telManager.getClass().getDeclaredMethod("getITelephony");
        getITelephonyMethod.setAccessible(true);             //私有化函数也能使用
        return (com.android.internal.telephony.ITelephony)getITelephonyMethod.invoke(telManager);
    }
}
```

(6) 新建源码包。为了使用源码中的方法,需要新建和源码中同样的包。在 src 目录上,右键单击"新建|包"选项,在弹出的窗口中新建一个名为 com.android.internal.telephony 的包,如图 9-11 所示。

在该包中添加文件 ITelephoney.aidl,然后将 Android 源码中的 ITelephony.aidl 复制到该新建文件中。可以在线查看 Android 源码,地址为 http:www.google.com/codesearch/p?hl=en#cZwlSNS7aEw/。在该网页中搜索 ITelephoney.aidl,即可找到该文件。

同理,继续添加 com.android.telephony 包,并添加文件 NeighboringCellInfo.aidl。添加该文件后,如果文件 ITelephony 中出现了 import 包错误,则在 android.telephony.NeighboringCellInfo 前添加"com.",修改为 import com.android.telephony.NeighboringCellInfo;成功添加包的效果如图 9-12 所示。

(7) 权限声明与广播注册。要获取电话的状态,应用程序必须具有相应的权限。而对于常驻广播接收器来实现获取电话状态,也需要在 AndroidManifest.xml 中进行注册。代

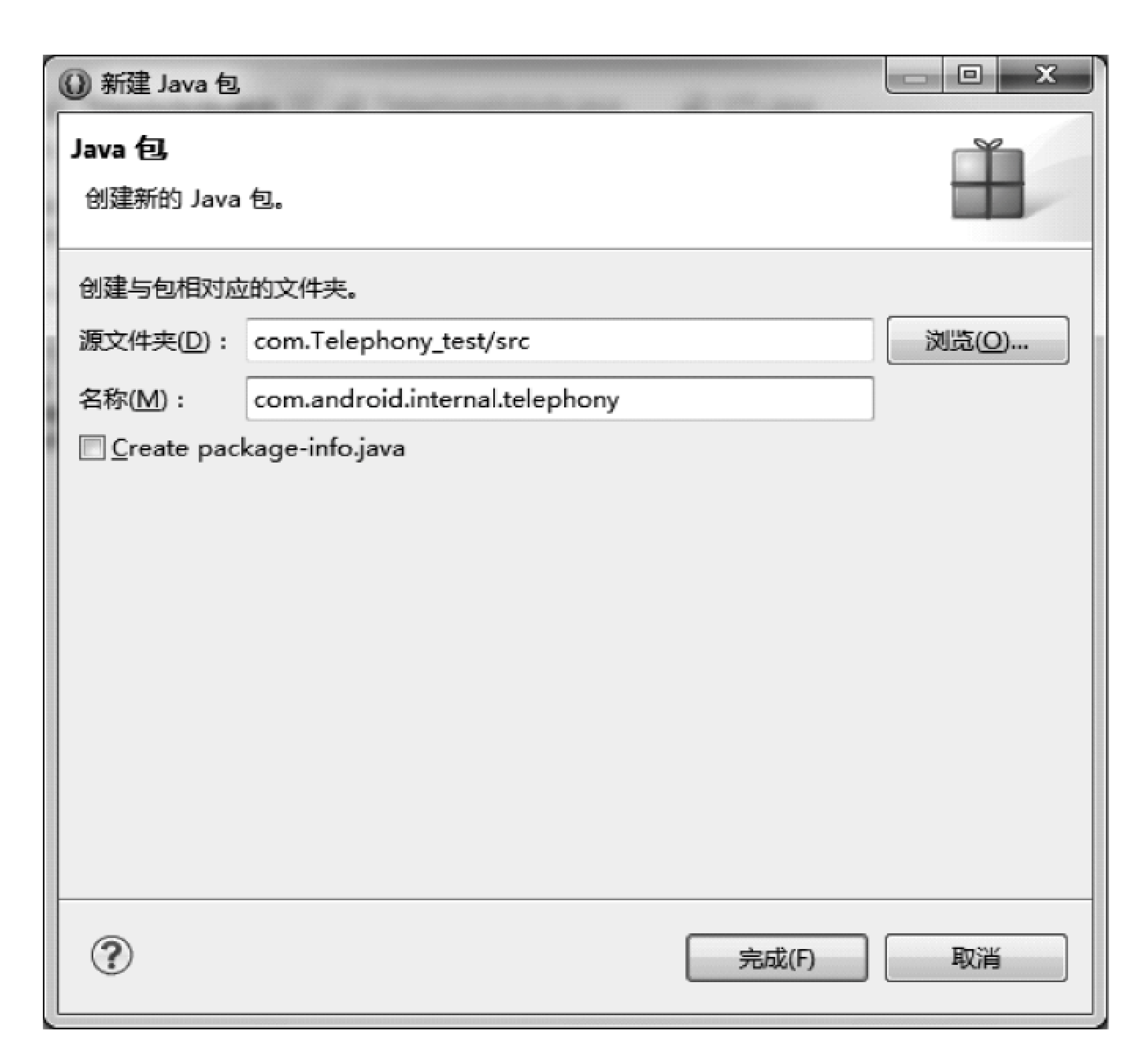


图 9-11　新建包



图 9-12　添加源码包

码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "fs.telephony_test"
    android:versionCode = "1"
    android:versionName = "1.0" >
    < uses - sdk
        android:minSdkVersion = "8"
        android:targetSdkVersion = "18" />
    <! -- 获取电话状态 -- >
    < uses - permission android:name = "android.permission.READ_PHONE_STATE" />
    <! -- 拨出电话 -- >
    < uses - permission android:name = "android.permission.PROCESS_OUTGOING_CALLS" />
    <! -- 电话 -- >
```

```
        <uses-permission android:name="android.permission.CALL_PHONE" />
        <application
            android:icon="@drawable/ic_launcher"
            android:label="@string/app_name" >
            <receiver
                android:name=".TelephoneActivity"
                android:priority="10000" >
                <intent-filter>
                    <action android:name="android.intent.action.PHONE_STATE" />
                </intent-filter>
            </receiver>
        </application>
    </manifest>
```

9.3.2 Android 短信

短信作为当今人和人交流中非常重要的方式，珍藏着大家不同时期的心情和成长。

1. 发送短信

Android 中短信主要采用 SmsManager 的 sendTextMessage()方法来发送文字短信，sendTextMessage()方法有 5 个参数，第 1 个参数为对方的手机号码(不能为空)，第 2 个参数为发送方的手机号号码(可以为空)，第 3 个参数为发送的短信内容(不能为空)，第 4 个参数为 PendingIntent 对象，用于判断发送短信是否成功(可以为空)，第 5 个参数也为 PendingIntent 对象，当用户接收到短信时会返回该对象(可以为空)。

【例 9-11】 下面通过一个案例来演示手机的发送短信功能。其具体实现步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Send_Message。

(2) 打开 res\layout 目录下的 main.xml 布局文件，在文件中声明两个 TextView 控件、两个 EditText 控件及一个 Button 控件。代码为：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:background="#ffcc66">
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="号码"/>
<!-- 电话号码输入 -->
<EditText
    android:id="@+id/et_phone"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"/>
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="短信"/>
```

```
<! -- 短信内容编辑 -->
<EditText
    android:id="@+id/et_content"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:minLines="3"
    android:gravity="left"/>
    <! -- 可显示 3 行 -->
    <! -- 设置左边输入 -->
<Button
    android:id="@+id/bt_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="发送"/>
</LinearLayout>
```

(3) 打开 src\fs.send_message 包下的 MainActivity.java 文件，在文件中编写发送短信。代码为：

```
package fs.send_message;
import java.util.List;
import android.app.Activity;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends Activity {
    private EditText mobileText;
    private EditText contentText;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //获取电话号文本框
        mobileText = (EditText)this.findViewById(R.id.et_phone);
        //获取短信内容文本框
        contentText = (EditText)this.findViewById(R.id.et_content);
        //获取按钮
        Button button = (Button)this.findViewById(R.id.bt_send);
        button.setOnClickListener(new View.OnClickListener(){
            public void onClick(View v) {
                //获取电话号码
                String moblie = mobileText.getText().toString();
                //短信内容
                String content = contentText.getText().toString();
                //获取短信管理器
                SmsManager smsManager = SmsManager.getDefault();
```

```
                    //如果汉字大于 70 个
                    if(content.length() > 70){
                        //返回多条短信
                        List<String> contents = smsManager.divideMessage(content);
                        for(String sms:contents){
                            smsManager.sendTextMessage(moblie,null,sms,null,null);
                        }
                    }else{
                    smsManager.sendTextMessage(moblie,null,content,null,null);
                        }
        Toast.makeText(MainActivity.this,"发送成功!",Toast.LENGTH_LONG).show();
                    }
                });
    }
}
```

(4) 打开 AndroidManifest.xml 文件配置项目,添加发送短信权限。代码为:

```
…
<uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />
  <!-- 添加短信服务 -->
   <uses-permission android:name="android.permission.SEND_SMS"/>
…
```

运行程序,发送短信界面如图 9-13 所示。



图 9-13　发送短信界面

2. 接收短信

除了从 Android 应用程序发送 SMS 消息外，还可以在应用程序中使用 BroadcastReceiver 对象接收传入的 SMS 消息。如果希望应用程序在收到一条特定的 SMS 消息时执行一个动作，这是很有用的，可能根据追踪的手机位置以防丢失或被盗。在这种情况下，可以编写一个应用程序，用来自动侦听包含一些秘密代码的 SMS 消息。一旦收到此类信息，即可给发送者发回一条包含位置坐标的 SMS 消息。

下面通过一个实例来演示在 Android 中实现手机短信的接收。

【例 9-12】 接收短信功能。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Receive_SMS。

(2) 打开 AndroidManifest.xml 文件，设置接收短信权限。

```
…
    </application>
     <!-- 设置短信声明权限 -->
     <uses-permission android:name="android.permission.RECEIVE_SMS">
     </uses-permission>
     <uses-permission android:name="android.permission.SEND_SMS"/>
</manifest>
…
```

(3) 打开 res\layout 目录下的 main.xml 文件，用于声明两个 TextView 控件、两个 EditText 控件和一个 Button 控件。代码为：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical"
android:background="@drawable/kp">
     <TextView
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:text="发送者号码"/>
    <EditText
          android:id="@+id/txtPhoneNo"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"/>
    <TextView
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:text="信息内容"/>
    <EditText
          android:id="@+id/txtMessage"
          android:layout_width="fill_parent"
          android:layout_height="150px"
          android:gravity="top"/>
    <Button
          android:id="@+id/btnSendSMS"
```

```
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "回复短信"/>
</LinearLayout>
```


(4) 打开 src\fs. receive_sms 包下的 MainActivity. java 文件,用于实现短信的接收。代码为:

```
package fs.receive_sms;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.widget.Toast;
public class MainActivity extends BroadcastReceiver
{
    @Override
    public void onReceive(Context context, Intent intent)
    {
        //获取传入 SMS 信息
        Bundle bundle = intent.getExtras();
        SmsMessage[] msgs = null;
        String str = "";
        if (bundle != null)
        {
            //检索接收到的 SMS 消息
            Object[] pdus = (Object[]) bundle.get("pdus");
            msgs = new SmsMessage[pdus.length];
            for (int i = 0; i < msgs.length; i++){
                msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]);
                str += "SMS from " + msgs[i].getOriginatingAddress();
                str += " :";
                str += msgs[i].getMessageBody().toString();
                str += "\n";
            }

            //显示新的 SMS 消息
            Toast.makeText(context, str, Toast.LENGTH_SHORT).show();
            //MainActivity 的扩展
            Intent mainActivityIntent = new Intent(context, MainActivity.class);
            mainActivityIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            context.startActivity(mainActivityIntent);
            //发送一个广播意图来更新活动中接收到的 SMS
            Intent broadcastIntent = new Intent();
            broadcastIntent.setAction("SMS_RECEIVED_ACTION");
            broadcastIntent.putExtra("sms", str);
            context.sendBroadcast(broadcastIntent);
        }
    }
}
```

运行程序，效果如图 9-14 所示。



图 9-14　短信接收界面

3. 短信群发

群发短信是一个十分实用的功能，逢年过节，很多人都喜欢通过群发短信向自己的朋友表示祝福。群发短信可以将一条短信同时向多个人发送。群发短信的实现十分简单，只要让程序遍历每个收件人号码并依次向每个收件人发送短信即可。

下面通过一个案例来演示 Android 手机的群发短信。

【例 9-13】 实现群发短信。其具体操作步骤为：

(1) 在 Eclipse 中创建一个 Android 应用项目，命名为 Group_Message。

(2) 打开 res\layout 目录下的 main. xml 文件，在文件中声明两个 TextView 控件、两个 EditText 控件及两个 Button 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#ffcc66">
< TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "号码(可选多个号码)"/>
< EditText
    android:id = "@ + id/numbers"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:editable = "false"
    android:cursorVisible = "false"
    android:lines = "2"/>
< TextView
```

```
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "短信"/>
    <EditText
        android:id = "@ + id/content"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:lines = "2"/>
    <LinearLayout
        android:orientation = "horizontal"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:gravity = "center_horizontal">
    <Button
        android:id = "@ + id/select"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "选择"/>
    <Button
        android:id = "@ + id/send"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "发送"/>
    </LinearLayout>
    </LinearLayout>
```

(3) 在 res\layout 目录下创建一个 list.xml 文件，用于实现列表选择多个号码。代码为：

```
<?xml version = "1.0" encoding = "UTF - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
<ListView
    android:id = "@ + id/list"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"/>
</LinearLayout>
```

(4) 打开 src\group_message 包下的 MainActivity.java 文件，在文件中实现短信群发。代码为：

```
package fs.group_message;
import java.util.ArrayList;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.PendingIntent;
import android.content.DialogInterface;
import android.content.Intent;
```

```
import android.database.Cursor;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.telephony.gsm.SmsManager;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.Toast;
public class MainActivity extends Activity
{
    EditText numbers,content;
    Button select,send;
    SmsManager sManager;
    //记录需要群发的号码列表
    ArrayList<String> sendList = new ArrayList<String>();
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        sManager = SmsManager.getDefault();
        //获取界面上的文本框、按钮组件
        numbers = (EditText) findViewById(R.id.numbers);
        content = (EditText) findViewById(R.id.content);
        select = (Button) findViewById(R.id.select);
        send = (Button) findViewById(R.id.send);
        //为send按钮的单击事件绑定监听器
        send.setOnClickListener(new OnClickListener()
        {
            public void onClick(View v)
            {
                for (String number : sendList)
                {
                    //创建一个PendingIntent对象
                    PendingIntent pi = PendingIntent.getActivity(
                        MainActivity.this,0,new Intent(),0);
                    //发送短信
                    sManager.sendTextMessage(number,null,content.getText().toString(),pi,
null);
                }
                //提示短信群发完成
                Toast.makeText(MainActivity.this,"短信群发完成",8000).show();
            }
        });
        //为select按钮的单击事件绑定监听器
```

```
            select.setOnClickListener(new OnClickListener()
            {
                @Override
                public void onClick(View v)
                {
                    //查询联系人的电话号码
                    final Cursor cursor = getContentResolver().query(
                        ContactsContract.CommonDataKinds.Phone.CONTENT_URI,null,null,null,null);
                    BaseAdapter adapter = new BaseAdapter()
                    {
                        @Override
                        public int getCount()
                        {
                            return cursor.getCount();
                        }
                        @Override
                        public Object getItem(int position)
                        {
                            return position;
                        }
                        @Override
                        public long getItemId(int position)
                        {
                            return position;
                        }
                        public View getView1(int position,View convertView,
                            ViewGroup parent)
                        {
                            cursor.moveToPosition(position);
                            CheckBox rb = new CheckBox(MainActivity.this);
                            //获取联系人的电话号码,并去掉中间的中划线
                            String number = cursor
                                .getString(
                                     cursor.getColumnIndex(ContactsContract.CommonDataKinds.
Phone.NUMBER))
                                .replace("-",""); 
                            rb.setText(number);
                            //如果该号码已经被加入发送人名单,默认选中该号码
                            if (isChecked(number))
                            {
                                rb.setChecked(true);
                            }
                            return rb;
                        }
                        @Override
                        public View getView(int arg0,View arg1,ViewGroup arg2) {
                            //TODO:自动存根法
                            return null;
                        }
                    };
                    //加载 list.xml 布局文件对应的 View
```

```
                    View selectView = getLayoutInflater().inflate(R.layout.list,
                        null);
                    //获取 selectView 中的名为 list 的 ListView 组件
                    final ListView listView = (ListView) selectView
                        .findViewById(R.id.list);
                    listView.setAdapter(adapter);
                    new AlertDialog.Builder(MainActivity.this)
                        .setView(selectView)
                        .setPositiveButton("确定")
                        new DialogInterface.OnClickListener()
                        {
                            @Override
                            public void onClick(DialogInterface dialog,
                                int which)
                            {
                                //清空 sendList 集合
                                sendList.clear();
                                //遍历 listView 组件的每个列表项
                                for (int i = 0; i < listView.getCount(); i++)
                                {
                                    CheckBox checkBox = (CheckBox) listView
                                        .getChildAt(i);
                                    //如果该列表项被选中
                                    if (checkBox.isChecked())
                                    {
                                        //添加该列表项的电话号码
                                        sendList.add(checkBox.getText()
                                            .toString());
                                    }
                                }
                                numbers.setText(sendList.toString());
                            }
                        }).show();
                }
            });
        }
        //判断某个电话号码是否已在群发范围内
        public boolean isChecked(String phone)
        {
            for (String s1 : sendList)
            {
                if (s1.equals(phone))
                {
                    return true;
                }
            }
            return false;
        }
    }
```

(5) 打开 AndroidManifest.xml 文件，在文件中添加短信群发权限。代码为：

```
…
    </application>
    <! -- 添加群发短信权限 -->
<uses-permission android:name="android.permission.READ_CONTACTS"></uses-permission>
<uses-permission android:name="android.permission.SEND_SMS"></uses-permission>
</manifest>
```

运行程序,效果如图 9-15 所示。



图 9-15 短信群发

程序中提供了一个带列表的对话框供用户选择群发 SMS 的收件人号码,程序则使用了一个 ArrayList<String>集合来保存所有的收件人号码。为了实现群发 SMS 功能,程序使用循环遍历 ArrayList<String>中的每个号码,并使用 SmsManager 依次向每个号码发送短信即可。

9.3.3 Android 电子邮件

与 SMS 消息传递类似,Android 还支持电子邮件。Android 上的 Gamil/Email 应用程序可以使用 POP3 或 IMAP 来配置电子邮件账户。除了使用 Gamil/Email 应用程序发送和接收电子邮件外,还可以通过编程方式从 Android 应用程序中发送电子邮件。

下面通过一个实例来演示在 Android 手机怎样实现电子邮件发送。

【例 9-14】 电子邮件发送。其具体实现步骤为:

(1) 在 Eclipse 中创建一个 Android 应用项目,命名为 Email_test。

(2) 打开 res\layout 目录下的 main.xml 布局文件,在文件中定义一个 Button 控件。代码为:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = "#aaccff">
<Button
    android:id = "@ + id/btnSendEmail"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "发送邮件" />
</LinearLayout>
```

(3) 打开 src\fs.email_test 包下的 MainActivity.java 文件，在文件中实现账号创建及发送邮件。代码为：

```
package fs.email_test;
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
public class MainActivity extends Activity {
    Button btnSendEmail;
    /** 第 1 次调用 activity 活动 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        btnSendEmail = (Button) findViewById(R.id.btnSendEmail);
        btnSendEmail.setOnClickListener(new View.OnClickListener()          {
            public void onClick(View v)
            {
                String[] to = {"Kemmling@learn2develop.net","Kemmling@gmail.com"};
                String[] cc = {"course@learn2develop.net"};
                sendEmail(to,cc,"Hello","Hello my friends!");
            }
        });
    }
    //--- 发送 SMS 消息到另一个装置 ---
    private void sendEmail(String[] emailAddresses,String[] carbonCopies,
    String subject,String message)
    {
        Intent emailIntent = new Intent(Intent.ACTION_SEND);
        emailIntent.setData(Uri.parse("mailto:"));
        String[] to = emailAddresses;
        String[] cc = carbonCopies;
        emailIntent.putExtra(Intent.EXTRA_EMAIL,to);
        emailIntent.putExtra(Intent.EXTRA_CC,cc);
        emailIntent.putExtra(Intent.EXTRA_SUBJECT,subject);
```

```
            emailIntent.putExtra(Intent.EXTRA_TEXT,message);
            emailIntent.setType("message/rfc822");
            startActivity(Intent.createChooser(emailIntent,"Email"));
        }
    }
```

运行程序,默认效果如图 9-16(a)所示,单击界面中的"发送邮件"按钮,效果如图(b)所示,设置相应的账号及密码单击"下一步"即实现邮箱创建,效果如图(c)所示。



(a) 默认界面　　(b) 账户设置　　(c) 创建账户

图 9-16　发送邮件

9.4　Android 定位

Android 支持 GPS 和网络地图,通常将各种不同的定位技术称为 LBS。LBS 是基于位置的服务(Location Based Service)的简称,它是通过电信移动运营商的无线电通信网络(如 GSM 网、CDMA 网)或外部定位方式(如 GPS)获取移动终端用户的位置信息(地理坐标或大地坐标),在地理信息系统(Geographic Information System,GIS)平台的支持下,为用户提供相应服务的一种增值业务。

在实现 GPS 定位前,先了解一下 GPS 的部分特性:

(1) GPS 定位需要依靠 3 颗或 3 颗以上的卫星。

(2) GPS 定位受环境影响较大,在晴朗的空地上,较容易搜索到卫星,而在室内通常是无法搜索到卫星的。

(3) GPS 定位需要使用 GPS 功能模块,而 GPS 功能模块的耗电量是巨大的。

在 Android 系统中,实现 GPS 定位的思路应该是:

(1) 获取 GPS 的 Location Provider。

(2) 将此 Provider 传入到 requestLocationUpdates()方法,让 Android 系统获知搜索位置方式。

(3) 实现了 GpsStatus. Listener 接口的对象,重写 onGpsStatusChanged()方法,向

LocationManager 添加次监听器，检测卫星状态（可选步骤）。

1. GPS 状态

在 Android 中提供了对应方法用于检查 GPS 的状态。

（1）获取首次定位时间

在 Android 中提供了 getTimeToFirstFix 方法用于获取 GPS 的首次定位时间。getTimeToFirstFix 方法用于获取最新的 GPS 引擎，当重新启动可获取首次定位所需的时间，其单位为毫秒（ms）。

注意：在空旷的地方，GPS 定位时间会较短；而障碍物较多的地方，定位时间会较长。

getTimeToFirstFix 方法的调用格式为：

```
public int getTimeToFirstFix()
```

（2）获取最大卫星数量

在 Android 中提供了 getMaxStatellites 方法用于获取在卫星列表中可返回的最大卫星数目。这个数目是 getStatellites 方法能够得到的最大卫星数目，但并不代表当前实际获得的卫星数量。其一般返回值为 255。

getMaxStatellites 方法的调用格式为：

```
public int getMaxStatellites()
```

（3）获取 GPS 卫星状态

在 Android 中提供了 getStatellites 方法用于获取 GPS 引擎的当前状态，该方法返回一个为 GpsStatellites 的对象数组值，其中包含了卫星列表。

getStatellites 方法的调用格式为：

```
public Iterable<GpsSatellite> getSatellites()
```

2. GPS 位置信息

在 Android 中提供了相关函数用于获取 GPS 位置信息，包括精度、方位、经纬度、海拔、速度、高度、运营商收费等信息。

（1）精度

在 Android 中提供了 getAccuracy 方法用于获取当前定位数据的精确度信息，其返回值为 float 类型的数据，单位为米（m）。精度反映了经度与纬度在定位上的误差。getAccuracy 的调用方法为：

```
Public float getAccuracy()
```

（2）方位

在 Android 中提供了 getBearing 方法用于获取 GPS 卫星的方位，其返回值为 float 类型。方位以地理北为参考，取值范围为 0°～360°。getBearing 的调用方法为：

```
Public float getBearing()
```

（3）高度

在 Android 中提供了 getAltitude 方法用于获取 GPS 卫星的高度，其返回值为 float 类型。getAltitude 的调用方法为：

```
Public float getAltitude()
```

(4) 经度

在 Android 中提供了 getLatitude 方法用于获取 GPS 卫星的经度,其返回值为 float 类型。getLatitude 的调用方法为:

```
public float getLatitude()
```

(5) 海拔

在 Android 中提供了 getAltitude 方法用于获取 GPS 卫星的海拔,其返回值为 float 类型。getAltitude 的调用方法为:

```
public float getAltitude()
```

(6) 纬度

在 Android 中提供了 getLongitude 方法用于获取 GPS 卫星的纬度,其返回值为 float 类型。getLongitude 的调用方法为:

```
Public float getLongitude()
```

(7) 速度

在 Android 中提供了 getSpeed 方法用于获取 GPS 卫星的速度,其返回值为 float 类型。getSpeed 的调用方法为:

```
Public float getSpeed()
```

3. GPS 参数

在 Android 中提供了相关函数用于获取 GPS 参数,包括方位角、高度角、伪随机数、信噪比等信息。

(1) 方位角

在 Android 中提供了 getAzimuth 方法用于获取方位角,其返回值为 true,方位角的取值范围为 0 度~360 度。getAzimuth 调用方法为:

```
public float getAzimuth()
```

(2) 高度角

在 Android 中提供了 getElevation 方法用于获取 GPS 卫星的高度角,其返回值为 true,高度角的取值范围为 0 度~360 度。getElevation 调用方法为:

```
public float getElevation()
```

(3) 伪随机数

在 Android 中提供了 getPrn 方法用于获取 GPS 卫星的伪随机数(PRN),其返回值为 int 类型。getPrn 的调用方法为:

```
public int getPrn()
```

(4) 信噪比

在 Android 中提供了 getSnr 方法用于获取 GPS 卫星的信噪比,其返回值为 float 类型。

getSnr 的调用方法为：

```
Public float getSnr()
```

下面通过一个实例来演示获取定位信息。

【例 9-15】 本实例通过手机实时获取定位信息，包括用户所在的经度、纬度、高度、方向、移动速度等。其具体实现步骤为：

（1）在 Eclipse 中创建一个 Android 应用项目，命名为 GPS_Data。

（2）打开 res\layout 目录下的 main.xml 文件，在文件中定义一个 EditText 控件。代码为：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:background = " # aabbcc">
< EditText
    android:id = "@ + id/show"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:editable = "false"
    android:cursorVisible = "false"/>
</LinearLayout >
```

（3）打开 src\fs.gps_data 包下的 MainActivity.java 文件，在文件中实现手机实时获取定位信息。代码为：

```
package fs.gps_data;
import android.app.Activity;
import android.content.Context;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.widget.EditText;
public class MainActivity extends Activity
{
    //定义 LocationManager 对象
    LocationManager locManager;
    //定义程序界面中的 EditText 组件
    EditText show;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //获取程序界面上的 EditText 组件
        show = (EditText) findViewById(R.id.show);
        //创建 LocationManager 对象
    locManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
        //从 GPS 获取最近的定位信息
        Location location = locManager.getLastKnownLocation(
            LocationManager.GPS_PROVIDER);
        //根据 EditText 的显示使用 location
```

```
        updateView(location);
        //设置每 3 秒获取一次 GPS 的定位信息
        locManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,3000,8,new LocationListener()
        {
            @Override
            public void onLocationChanged(Location location)
            {
                //当 GPS 定位信息发生改变时,更新位置
                updateView(location);
            }
            @Override
            public void onProviderDisabled(String provider)
            {
                updateView(null);
            }
            @Override
            public void onProviderEnabled(String provider)
            {
                //当 GPS LocationProvider 可用时,更新位置
                updateView(locManager
                    .getLastKnownLocation(provider));
            }
            @Override
            public void onStatusChanged(String provider,int status,
                Bundle extras)
            {
            }
        });
    }
    //更新 EditText 中显示的内容
    public void updateView(Location newLocation)
    {
        if (newLocation != null)
        {
            StringBuilder sb = new StringBuilder();
            sb.append("实时的位置信息: \n");
            sb.append("经度: ");
            sb.append(newLocation.getLongitude());
            sb.append("\n 纬度: ");
            sb.append(newLocation.getLatitude());
            sb.append("\n 高度: ");
            sb.append(newLocation.getAltitude());
            sb.append("\n 速度: ");
            sb.append(newLocation.getSpeed());
            sb.append("\n 方向: ");
            sb.append(newLocation.getBearing());
            show.setText(sb.toString());
        }
        else
        {
            //如果传入的 Location 对象为空则清空 EditText
            show.setText("");
        }
    }
}
```

(4) 为程序添加 GPS 信号的访问权限,打开 AndroidManifest.xml 文件,添加权限代码为:

```
…
    </application>
    <!-- 授权获取定位信息 -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
</manifest>
```

运行程序，打开 DDMS 的 Emulator Control 面板，如图 9-17 所示，填写相关数据，并单击界面中的 Send 按钮，在 Android 中即可接收到定位信息，效果如图 9-18 所示。

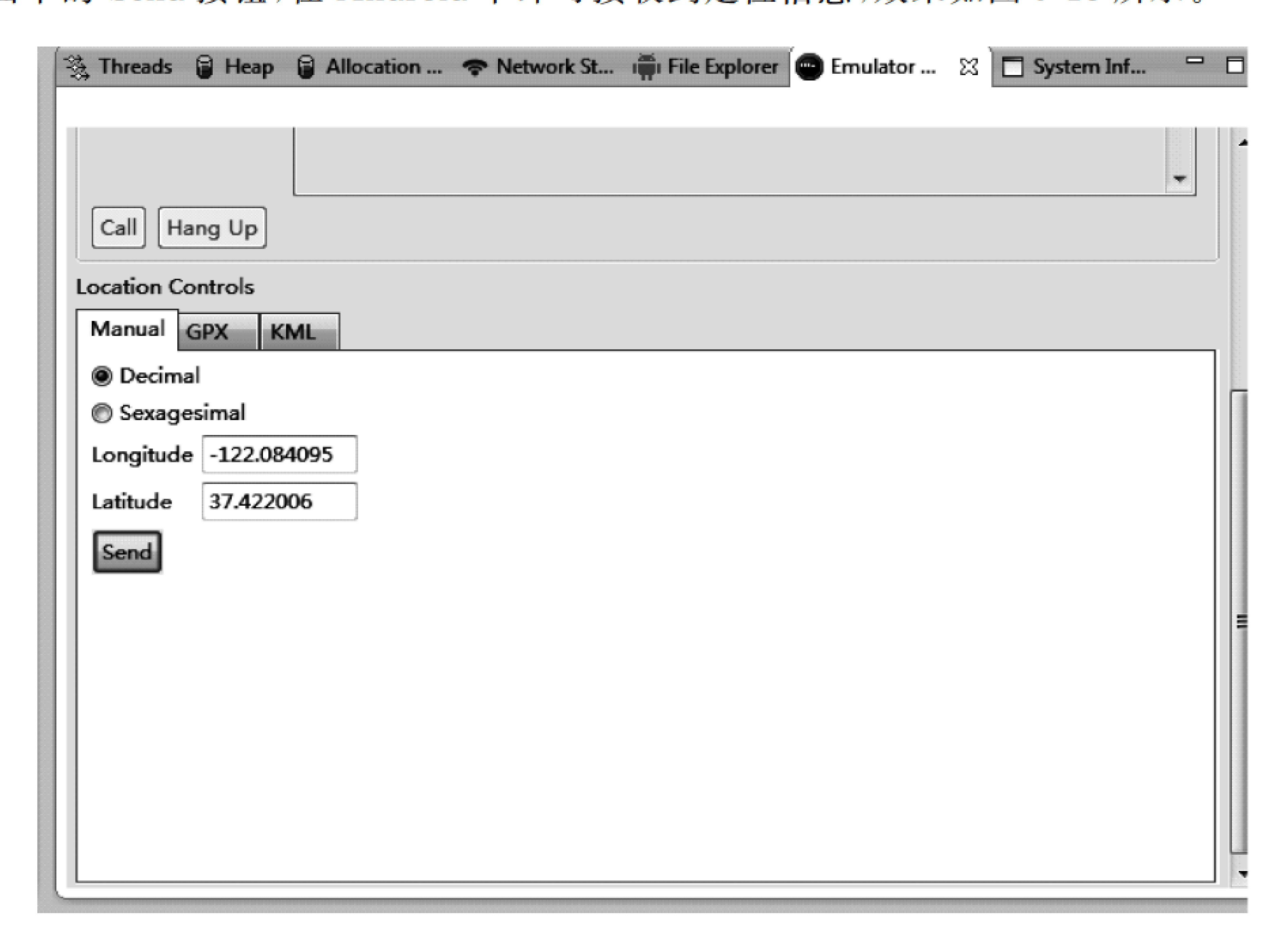


图 9-17　Emulator Control 面板



图 9-18　实时的定位信息

参 考 文 献

[1] 张余. Android 网络开发从入门到精通. 北京：清华大学出版社，2014.
[2] 李刚. 疯狂 Android 讲义. 北京：电子工业出版社，2012.
[3] 李佐彬. Android 开发入门与实战体验. 北京：机械工业出版社，2011.
[4] 软件开发技术联盟. Android 开发实战. 北京：清华大学出版社，2013.
[5] 欧阳零. Android 编程兵书. 北京：电子工业出版社，2014.
[6] 明日科技. Android 从入门到精通. 北京：清华大学出版社，2012.
[7] 楚无咎. Android 编程经典 200 例. 北京：电子工业出版社，2013.
[8] 李波，史江萍，王祥凤. Google Android 4. X 从入门到精通. 北京：清华大学出版社，2012.
[9] Wei-Meng Lee. Android 编程入门经典. 何晨光，李洪刚，等译. 北京：清华大学出版社，2012.
[10] 高洪岩. Android 学习精要. 北京：清华大学出版社，2012.
[11] http://blog. csdn. net/wangkuifeng0118/article/details/7339578.
[12] http://blog. csdn. net/ithomer/article/details/7281041.
[13] http://blog. csdn. net/geniusxiaoyu/article/details/7305270.
[14] http://www. cnblogs. com/linjiqin/archive/2011/05/26/2059133. html.
[15] http://blog. csdn. net/alex0203/article/details/7081880.
[16] http://www. cnblogs. com/TerryBlog/archive/2010/07/09/1774475. html.
[17] http://www. cnblogs. com/hexiaochun/archive/2012/10/08/2715832. html.
[18] http://www. 2cto. com/kf/201111/110955. html.
[19] http://blog. sina. com. cn/s/blog_8373f9b501015fni. html.
[20] http://blog. csdn. net/beyond0525/article/details/8835262.
[21] http://blog. csdn. net/lxw1980/article/details/6031978.
[22] http://www. pipaw. com/androidcourse/news_3136. html.
[23] http://book. 2cto. com/201210/6245. html.
[24] http://blog. csdn. net/lee576/article/details/7900228.
[25] http://blog. csdn. net/lee576/article/details/7900228.
[26] http://www. 2cto. com/kf/201307/226948. html.
[27] http://blog. csdn. net/tianjf0514/article/details/7557383.
[28] http://blog. sina. com. cn/s/blog_78e3ae430100py4p. html.
[29] http://blog. csdn. net/aikongmeng/article/details/14129319.
[30] http://edison-cool911. iteye. com/blog/759355.
[31] http://www. cnblogs. com/happyDays/archive/2013/08/01/3230980. html.
[32] http://byandby. iteye. com/blog/828799.
[33] http://livehappy. iteye. com/blog/1004399.
[34] http://blog. csdn. net/wangkuifeng0118/article/details/7044195.
[35] http://www. jb51. net/article/42123. htm.
[36] http://www. cnblogs. com/skiz/archive/2012/10/26/2741649. html.
[37] http://liangruijun. blog. 51cto. com/3061169/647456/.
[38] http://blog. csdn. net/mayingcai1987/article/details/6232852.

[39] http://www.cnblogs.com/linjiqin/archive/2011/02/23/1962535.html.
[40] http://blog.csdn.net/jianghuiquan/article/details/8350524.
[41] http://blog.csdn.net/jianghuiquan/article/details/8350525.
[42] http://blog.csdn.net/jianghuiquan/article/details/8350519.
[43] http://www.oschina.net/code/snippet_12_5246.
[44] http://www.cnblogs.com/GnagWang/archive/2010/11/26/1888762.html.
[45] http://baike.baidu.com/subview/1241829/9322617.htm? fr=aladdin.
[46] http://blog.csdn.net/yihui823/article/details/6702273.
[47] http://www.2cto.com/kf/201111/110955.html.
[48] http://blog.csdn.net/xinem/article/details/7083523.
[49] http://www.cnblogs.com/playing/archive/2011/04/07/2008228.html.
[50] http://www.cnblogs.com/salam/archive/2010/10/06/1844703.html.
[51] http://aina-hk55hk.iteye.com/blog/675377.
[52] http://www.oschina.net/code/snippet_54100_7440.
[53] http://blog.csdn.net/x605940745/article/details/9703845.
[54] http://blog.csdn.net/x605940745/article/details/9704229.